

**UNIVERSITÀ DI PISA**  
**Scuola di Dottorato in Ingegneria "Leonardo da Vinci"**



**Corso di Dottorato di Ricerca in  
Ingegneria dell'Informazione**

**Tesi di Dottorato di Ricerca**

# **HIDE: User centred Domotic evolution toward Ambient Intelligence**

***Dario Russo***

*Anno 2015*



**UNIVERSITÀ DI PISA**

**Scuola di Dottorato in Ingegneria "Leonardo da Vinci"**



**Corso di Dottorato di Ricerca in  
Ingegneria dell'Informazione**

**Tesi di Dottorato di Ricerca**

# **HIDE: User centred Domotic evolution toward Ambient Intelligence**

*Autore:*

*Dario Russo*

*Relatori:*

*Prof. Stefano Giordano*

*Dott. Vittorio Miori*

*Anno 2015*

**SSD: ING-INF/03 - TELECOMUNICAZIONI**



---

## Sommario

La realizzazione degli obiettivi di ricerca del *Pervasive computing* e dell'*Ambient Intelligence* (Aml) è ancora lontana dall'essere raggiunta soprattutto negli ambiti della Domotica e delle relative applicazioni domestiche. Secondo la concezione dell'*Ambient Intelligence*, le più avanzate tecnologie sono quelle che scompaiono: ciò significa che la tecnologia all'interno delle nostre case, dovrebbe essere integrata nell'ambiente e diventare invisibile. L'intero ambiente fisico in cui gli utenti saranno immersi dovrà essere fornito di un sistema informatico nascosto, dotato di software appropriato e in grado di mostrare un comportamento intelligente. Recentemente sono iniziate a comparire varie implementazioni di questo tipo, ma poche di queste sono dedicate alla casa o agli ambienti di vita in generale. Una probabile causa di ciò, condivisa in letteratura, è la segmentazione dei numerosi standard e delle soluzioni proprietarie domotiche che distorcono il mercato, con la conseguenza di un'offerta scarsa di dispositivi e presenza di sistemi non interoperabili.

L'obiettivo di questo lavoro di ricerca è quello di proporre una possibile soluzione a tale problema, da un lato con la realizzazione di un sistema software progettato per rendere eterogenei ed interoperabili i sistemi domotici nativamente incompatibili tra loro, e dall'altro con l'ideazione e al messa in opera di un'applicazione software in grado di apprendere il comportamento e le abitudini degli abitanti. Tale lavoro contribuisce attivamente all'aumento del comfort e della sicurezza, anticipando e cercando di prevenire possibili esigenze o situazioni di emergenza per la salute. Applicando tecniche di riconoscimento delle attività e di apprendimento automatico, il lavoro di ricerca si concentra sulla messa in opera di un'applicazione che riesce ad apprendere i comportamenti e le abitudini dell'utente, al fine di migliorare la sua qualità di vita. La soluzione proposta opera in un ambiente tecnologicamente arricchito, come ad esempio una casa domotica o un edificio intelligente. L'applicazione realizzata, oltre a rendere la vita più comoda per gli utenti normodotati, rappresenta una grande facilitazione per contribuire ad aumentare l'autonomia e la sicurezza agli occupanti disabili o anziani e in particolare a chi soffre di malattie croniche o è gravemente malato.

Il prototipo è stato sviluppato ed è attualmente funzionante presso il laboratorio di Domotica del CNR di Pisa, dove è stato fedelmente ricreato un vero ambiente domestico.



---

## Abstract

Pervasive Computing and Ambient Intelligence (Aml) visions are still far from being achieved, especially with regard to Domotics and home applications. According to the vision of Ambient Intelligence (Aml), the most advanced technologies are those that disappear: at maturity, computer technology should become invisible. All the objects surrounding us must possess sufficient computing capacity to interact with users, the surroundings and each other. The entire physical environment in which users are immersed should thus be a hidden computer system equipped with the appropriate software in order to exhibit intelligent behavior. Even though many implementations have started to appear in several contexts, few applications have been made available for the home environment and the general public. This is mainly due to the segmentation of standards and proprietary solutions, which are currently confusing the market with a sparse offer of uninteroperable devices and systems. Although modern houses are equipped with smart technological appliances, still very few of these appliances can be seamlessly connected to each other.

The objective of this research work is to take steps in these directions by proposing, on the one hand, a software system designed to make today's heterogeneous, mostly incompatible domotic systems fully interoperable and, on the other hand, a feasible software application able to learn the behavior and habits of home inhabitants in order to actively contribute to anticipating user needs, and preventing emergency situations for his health. By applying machine learning techniques, the system offers a complete, ready-to-use practical application that learns through interaction with the user in order to improve life quality in a technological living environment, such as a house, a smart city and so on. The proposed solution, besides making life more comfortable for users without particular needs, represents an opportunity to provide greater autonomy and safety to disabled and elderly occupants, especially the critically ill ones.

The prototype has been developed and is currently running at the Pisa CNR laboratory, where a home environment has been faithfully recreated.

**Keywords:** Ambient Intelligence, Interoperability, Domotics, Internet of Things, Machine Learning, Semantics, E-health.





*This work is dedicated to my little son Diego and to his magic smile.*



---

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivations	1
1.2	Involved technologies	4
1.2.1	Ambient Assisted Living	4
	Assistive Technologies	5
	Independent Living	6
1.2.2	Ambient Intelligence	6
1.2.3	Data Mining	7
1.2.4	Digital ecosystem	9
1.2.5	Domotics	11
1.2.6	Internet Of Things	12
1.2.7	Semantic web and ontologies	13
	Development of Ontologies	14
	Ontology editors	18
1.2.8	Machine Learning	19
1.3	Thesis Structure	20
<b>2</b>	<b>State of the art</b>	<b>21</b>
2.1	Ambient Assisted Living	21
2.1.1	HERA	21
2.1.2	HOMEdotOLD	21
2.1.3	I2HOME	22
2.1.4	MPower	22
	Use case methodology	22
	Requirement Analysis	23
2.1.5	OASIS	24
	Use case methodology	24
	Requirement Analysis	24
2.1.6	PERSONA	25
	Use case methodology	26

	Requirement Analysis .....	28
2.1.7	SOPRANO .....	30
	Use case methodology .....	30
	Requirement Analysis .....	31
2.1.8	universAAL .....	32
	Use case methodology .....	34
	Requirement Analysis .....	35
2.2	Digital ecosystem .....	38
2.2.1	DBE project .....	39
	DBE architecture .....	39
	DBE infrastructural services .....	40
2.2.2	Meteor-S project .....	42
	The semantic .....	42
	The components .....	43
2.3	Interoperability using semantics .....	43
2.3.1	Introduction .....	43
2.3.2	eBiz .....	44
	Introduction .....	44
	The architecture .....	45
2.3.3	Kassetts .....	48
	Introduction .....	48
2.3.4	Moda-ML .....	49
2.4	Semantics for Domotics .....	50
2.4.1	DogOnt .....	50
2.4.2	SensorML .....	51
2.5	Other works .....	54
<b>3</b>	<b>Architecture .....</b>	<b>57</b>
3.1	Introduction .....	57
3.2	Dictionary .....	58
3.3	Analysis requirements .....	58
3.4	Use Case .....	59
3.5	Network architecture .....	61
	3.5.1 Domotic agents .....	63
	3.5.2 User agents .....	63
	3.5.3 Intelligent agents .....	64
3.6	Semantic approach .....	64
3.7	Implementation tools .....	65
	3.7.1 Java .....	65
	3.7.2 JADE .....	67
	3.7.3 OWLAPI .....	68

<b>4</b>	<b>Domotic agent</b>	71
4.1	Introduction	71
4.1.1	UPnP system	71
	Introduction	71
	UPnP technology	72
	The UPnP Device Architecture	72
4.1.2	KNX system	74
	Introduction	74
	KNX Group address	75
	KNX Physical address	75
	KNX Specifications	76
	KNX Tools	76
	Interesting implementations	77
4.2	Interactions with domotic bus	77
4.3	Semantic layer	79
4.4	IPv6 and IoT	81
4.4.1	Introduction	81
4.4.2	State of the art	83
4.4.3	The IPv6 gateway	84
4.4.4	The web interface	85
<b>5</b>	<b>User agent</b>	89
5.1	Introduction	89
5.2	Semantic layer	89
5.3	Track and identification of users	90
5.4	User interaction with the environment	92
<b>6</b>	<b>Intelligent agent</b>	97
6.1	Introduction	97
6.1.1	Associative rules	99
	Frequent Itemset Generation: Apriori	99
6.2	Agent that learns and anticipates user needs	100
6.2.1	Association rules manager	100
6.2.2	Statistical rules manager	104
6.3	Agent that anticipates user health emergencies	107
<b>7</b>	<b>Test and verification</b>	111
<b>8</b>	<b>Conclusions and future works</b>	115
8.1	Future works	116
	<b>References</b>	119



---

## List of Figures

2.1	MPower use case specifications . . . . .	23
2.2	OASIS use case specifications . . . . .	25
2.3	Use Case design in the Persona project . . . . .	26
2.4	Persona user requirements analysis process . . . . .	29
2.5	SOPRANO use case specifications . . . . .	31
2.6	The root concept map of the universAAL reference model . . . . .	34
2.7	The root concept map of the universAAL reference model . . . . .	35
2.8	UniversAAL use case specifications . . . . .	36
2.9	UniversAAL iterative process . . . . .	37
2.10	The eBiz architecture . . . . .	44
2.11	The eBiz architecture . . . . .	46
2.12	The eBiz use profile definition . . . . .	47
2.13	The eBiz recommended standards . . . . .	48
2.14	The DogOnt ontology . . . . .	51
2.15	The DogOnt architecture . . . . .	52
2.16	SensorML overview . . . . .	54
3.1	Hierarchy of actors . . . . .	59
3.2	Hierarchy of actors . . . . .	60
3.3	User use case diagram . . . . .	60
3.4	Use case diagram . . . . .	61
3.5	Agent class diagram . . . . .	62
3.6	Intelligent Agent is an Environmental Manager . . . . .	63
3.7	Ontological representation of the Lamp class, using UML . . . . .	65
3.8	JADE behaviour model, using UML . . . . .	68
3.9	Generic JADE agent architecture . . . . .	69
4.1	The ontologies used to represent devices . . . . .	77
4.2	The Domotic Agent reads an event from the bus . . . . .	78
4.3	A screenshot of the ontology that defines the Dimmerling Lamp . . . . .	80

4.4	A screenshot of the ontology that merges the DomOnt with knxDomOnt ontologies for the KNX Domotic Agent .....	81
4.5	Device callable functions .....	86
4.6	Device function call .....	87
4.7	Response of the device function call .....	87
5.1	Class hierarchy of the ontology .....	90
5.2	List of properties of the ontology .....	91
5.3	Web interface used to control the home environment: light .....	93
5.4	Web interface used to control the home environment: light with radio buttons .....	94
5.5	Interface for Android used to control the home environment .....	94
5.6	The "Pronto" remote control .....	95
6.1	Intelligent agent architecture .....	98
6.2	Modules of the Intelligent agent .....	101
6.3	The Apriori algorithm .....	101
6.4	Association rules manager life-cycle .....	103
6.5	Reinforcement algorithm .....	105
6.6	Example of medical recommendations .....	107
6.7	Creation of structures in Knowledge repository .....	108
6.8	Filled structures in Knowledge repository .....	108
7.1	A view of the ISTI-CNR laboratory .....	111



## Introduction

This work is the result of the research and the development of the activities that I have conducted within the *Socialize AAL* European project, the *Arianna* "Industria 2015 - Made in Italy" project, and *Shell* Italian Ministry of Education and of University project for the development and strengthening program of national technology clusters.

My activities were conducted at the *Domotics Lab* of the *Institute of Science and Technologies of Information (ISTI)* of the *National Research Council of Italy (CNR)* under the responsibility and supervision of Dr. Vittorio Miori, with the collaboration of the *TLC Networks Research Group* of the *Information Engineering Department of the University of Pisa* under the responsibility and supervision of Prof. Stefano Giordano.

### 1.1 Motivations

*Ubiquitous Computing* and *Ambient Intelligence (Aml)* refer to a vision of the future information society where humans will be surrounded by intelligent interfaces supported by computing and networking technologies that will be everywhere and, largely thanks to the miniaturization of computer components, embedded in objects such as furniture, clothes, vehicles, roads and smart materials.

New advanced services will be created by exploiting the ability to make these objects interact with other people's objects and with the environment. Such technologies will be designed to be 'invisible' to people, who will use them without even realizing it. Thus, the actual computing capacity should remain in the background, in the periphery of our attention, and should only move to the center if and when necessary.

In the near future the home will be a technologically rich environment able to offer a wide range of network-based services through support middleware in gateways which will make them discoverable by and accessible to residential environments. The domestic network will connect all household appliances and displays regardless of manufacturer, and will moreover interact with the personal area network and the body network of each person in the home. Within homes, local networks with potentially different underlying implementations can be combined, or at least managed, as one logical network. The

need to support interoperability services, such as bridges, gateways or adapters [63] between different networks in the backbone, as well as in access and local networks, is an essential prerequisite for all *AmI* applications. The *AmI* vision foresees that our homes will contain a distributed network of intelligent devices that can adapt themselves so as to satisfy and anticipate users' needs. Thus, *Ambient Intelligence* refers to the presence of a digital environment that is sensitive, adaptive and responsive to people's presence.

Unfortunately, the state of the art of the Weiser Ubiquitous Computing vision [88] is still far from becoming a reality, though research activities are continuously proposing new services, algorithms and perspectives [1] able to provide ever more powerful and sophisticated solutions.

If we focus on the home, *Ambient Intelligence* may be seen as the layer on top of the domotics, per se. The aim of *Ambient Intelligence* is to progress from the mere programming of isolated devices, to integrating them in order to achieve global, unified goals. Home networks, that is, the specific Local Area Networks for home environments, include both networks to the home (access networks) and networks throughout the home. The inter-working capabilities are required beyond house boundaries, towards external services and towards other houses as nodes of a global network.

The current immaturity of the field of domotics and, more specifically, the lack of definition of application requirements, have led to the development of a large number of ad hoc proposals, which unfortunately are often limited and difficult to integrate. In order to make the advent of genuine *AmI* applications possible, there is a crucial need to define and develop a standard way forward. Although domotic and smart home technologies are currently ready and operational, they have as yet not been able to garner a broad consumer market, some of the main causes being the lack of standards and interoperability, as well as the absence of any "must-have" new functionality provided at an appealing cost.

Today's e-health solutions provide important contributions to the health management of the elderly and chronically ill within their own homes. Indeed, they provide for constant monitoring of many vital parameters via portable sensors (inserted into shirts, bracelets, watches, etc.), in order to be able to identify and opportunely signal any hazardous situation requiring intervention. In most cases, however, by the time the call for help is issued, the emergency is already in progress. A system able to anticipate danger before life-threatening situations arise would certainly lead to a faster, and more effective intervention. The ability to anticipate and recognize certain behaviors or events heralding serious health problems in time can often save lives.

One interesting open issue regarding *Home Automation* and *Ambient Intelligence* is related to recognizing unusual or dangerous situations in order to anticipate health problems or special individual home user needs. Indeed, every user has different behaviors and habits that may depend on his preferences, inabilities or state of health. Such problems may be addressed by monitoring users' habitual activities, which enables creating rules-based profiles in order to capture and formalize their normal behavior. Such monitoring activities can be carried out using a system based on machine learning, which

exploits artificial intelligence algorithms to learn user habits and build a knowledge base that enables taking into account experiences accumulated during day-to-day activities. However, many people have privacy concerns about having their behaviors monitored [62], the rules for monitoring implementation and what information is transmitted and to whom.

To make a system able to recognize any abnormal behaviour pattern and to anticipate user needs, the system should be provided by a software able to learn the behaviour and habits of home inhabitants. This software should be an adaptive, context-aware application that thus works in a fully interoperable environment. In order to anticipate and recognize such situations, data on user activity and the surrounding environment need to be contextualized and enhanced semantically. This aim can be achieved by exploiting one of the most important paradigms underlying the *Web 3.0: Semantic Intelligence* [44]. The Semantic Web [11] is an Internet space that includes documents (or portions thereof) describing the explicit relations between things, and containing semantic information intended for automated processing by machines. Not only Web pages, but databases, programs, sensors and even household appliances will be able to present data, multimedia, and status information in forms that powerful computing agents can use to search, filter and prepare information in new and exciting ways. By way of definition then, in the Semantic Web an ontology [7] is a partial conceptualization of a given knowledge domain, shared by a community of users, that has been defined in a formal, machine-processable language for the explicit purpose of sharing semantic information across automated systems.

The objective of this research work is to take steps in the direction of *Internet Of Things* and *Ambient Intelligence*. The work proposes a feasible software application able on the one hand, to abstract the peculiarities of underlying heterogeneous and natively not interoperable domotic systems letting them co-exist and interwork, and on the other hand, to learn the habits and behaviours of people in their own residence in order to anticipate their needs, and hence to recognize any abnormal behaviour pattern signalling potential imminent health crises, preventing emergency situations.

By applying semantic and machine learning techniques, the solution offers a complete and ready-to-use practical application that learns through interaction with the user in order to improve the quality of life in a technological living environment, such as a house, a smart city and so on. The proposed solution is currently suitable both for application to comfort and health issues. The solution represents an opportunity to provide greater autonomy and safety to disabled and elderly occupants, especially the critically ill ones.

To this end, the system is endowed not only with knowledge of the devices and their functions (knowing what a lamp, thermostat or switch means), but also with tools that enable it to learn concepts such as room, furniture, object position and the repercussions (effects) of the use of such objects in the environment in order to make possible to assign to the system a specific goal to perform without however having to specify each single action making it up. For these purposes, the semantic layer provides solutions for:

- semantic interoperability among heterogeneous technological systems and devices, to mask their technological differences inside the framework;
- equipping the user environment, the domotic devices and their functionalities with semantic capabilities;
- defining the position of domotic devices and furniture in order to communicate with them using their location descriptions (e.g. the night-table lamp, the kitchen TV);
- modelling the effects within the environment caused by the domotic functions in order to understand the impact that a device will have on the surrounding area.

The prototype has been developed and is currently running at the Pisa CNR laboratory, where a home environment has been faithfully recreated.

### 1.2 Involved technologies

The research area is evolving its interests following new programming approaches. In particular, during these recent years, many efforts were made by the research community regarding *Ambient Assisted Living*, *Ambient Intelligence*, *Data Mining*, *Digital Ecosystem*, *Domotics*, *Internet Of Things*, *Machine Learning* and *Semantic web and ontologies* paradigms.

#### 1.2.1 Ambient Assisted Living

*Ambient Assisted Living* (AAL) aims to extend the time people can live independently in their home environment by increasing their autonomy and self-confidence, the discharge of monotonously everyday activities, to monitor and care for the elderly or ill persons, to enhance the security and to save resources.

*Ambient Assisted Living* (AAL) is an European Initiative based on the Article 169 of the European Treaty that was born in order to address the needs of the ageing population, to reduce innovation barriers of forthcoming promising markets and also to lower future social security costs. This initiative is planned to be implemented during the 7th EU Framework Programme. The overall goals of AAL are the following:

- improve the quality of life of elderly people in their homes as it is known that they prefer to live in their own home instead of living at an old people's home;
- reduce the costs associated with elderly care.

AAL addresses in particular the issues affecting an ageing population and targets the needs of the individuals as well as their caretakers, but two groups of persons are considered:

- elderly people who can be disabled or actively aged. Actively aged are those elderly people whose limitations because of age are not perceived as a disability;
- disabled people.

AAL is a European initiative but Europe is not the only one facing the problem of an ageing society because the problem is a worldwide one. The philosophy behind AAL, which is, assisting elderly people to live independently in their own home environment as long as possible, is extended worldwide. In the following paragraphs we will describe projects and initiatives that have been developed or are being developed not only in Europe but in the rest of the world too. AAL is not a technology but a philosophy. Typically, several technologies will be needed to develop AAL solutions. Those technologies include products and services that enable persons to perform tasks or functions at a level similar to an earlier experience, and/or contribute to a lifestyle of independence. Some of these AAL-related technologies are the following:

- software and network technologies;
- sensors and actuators;
- human machine interfaces;
- new materials;
- embedded systems;
- several technologies involved in smart homes;
- other ambient intelligence technologies.

### **Assistive Technologies**

*Assistive Technologies* are those products and services that enable persons (regardless of age) to perform a function that due to some disability would otherwise be difficult to perform.

While some disabilities are associated with ageing, ageing itself is not a disability and even if some limitations exist many elderly people do not perceive themselves as being disabled. On the one hand this means that assistive technologies would be a part of AAL technologies. On the other hand, healthcare technologies are not AAL technologies but are closely related to them because ageing population is the most concerned with health-care. AAL technologies cover a wider range of ageing-related concerns. They represent a new category of technologies with the following functionalities:

- enable elderly and disabled people to live in their own environment;
- assist ageing people to function in society, facilitating social contacts, in addition to context-based infotainment and entertainment;
- provide communication services to ageing persons so that they can communicate with family, caregivers and medical personnel;
- tele-health, which means allowing a medical source to remotely monitor, diagnose and treat a patient.

Each functionality describes functions that may apply less to persons of a certain age than to “quality of life” and independence regardless of age.

### Independent Living

The concept of *Independent Living (IL)* challenges the preconceived medial models of disability and old age, by emphasizing self-determination and equal opportunities and the removal of societal barriers to participation. The medical view casts frail older and presents disabled people as defective and deviant, a burden for society and as passive recipients of professional interventions. *IL* emphasizes active social participation and the organization of societal supports within a radical agenda. Recent disability rights legislation and the recasting of health and social care policy has begun to move in this direction, but older and disabled people still find themselves marginalized within society. A review of the *IL* concept is provided in the paper by Sarah Gillinson, Hannah Green and Paul Miller [37].

#### 1.2.2 Ambient Intelligence

In computing, *Ambient Intelligence (AmI)* refers to electronic environments that are sensitive and responsive to the presence of people. With the increasing influence of technology on our lives, it is becoming ever more important to offer new ambient intelligence solutions which enable humans to adapt and organize their lives around computational technologies able to adapt to meet user behaviour. An important concept related to *AmI* is *Internet Of Things*. In computing, the *Internet Of Things*, also known as the *Internet Of Objects*, refers to the direct network interconnection with all our everyday objects. It is described as a self-configuring wireless network of sensors the purpose of which would be to interconnect all things. *Ambient Intelligence* combined with *Internet Of Things* permits easily and with more efficiency, to carry out people's everyday life activities in a natural way.

The characteristics of an *Ambient Intelligence* are to be:

- *embedded*: a network of devices that are integrated into the environment;
- *context aware*: environmental devices able to recognize the user and the context around him;
- *personalized*: the environment is tailored to the user's needs;
- *adaptive*: the environment change in response to the user;
- *anticipatory*: the environment anticipates user's desires without conscious mediation;
- *hidden*: users must not notice an high technology intervention. The technology must disappear into the user's surroundings.

*Ambient Intelligence* should also:

- facilitate human contacts;
- be oriented towards community and cultural enhancement;
- inspire trust and confidence;
- be consistent, sustainable, personal, societal and environmental;
- be made easy to control by ordinary people.

### 1.2.3 Data Mining

*Data Mining*, the extraction of hidden predictive information from large databases, is a powerful new technology with great potential to help companies focus on the most important information in their data warehouses. *Data Mining* tools predict future trends and behaviours, allowing businesses to make proactive, knowledge-driven decisions. The automated, prospective analyses offered by *Data Mining* move beyond the analyses of past events provided by retrospective tools typical of decision support systems. *Data Mining* tools can answer business questions that traditionally were too time consuming to be solved. They scour databases for hidden patterns, finding predictive information that experts may miss because it lies outside their expectations.

Most companies already collect and refine massive quantities of data. *Data Mining* techniques can be implemented rapidly on existing software and hardware platforms to enhance the value of existing information resources, and can be integrated with new products and systems as they are brought on-line. When implemented on high performance client/server or parallel processing computers, *Data Mining* tools can analyse massive databases to deliver answers to questions such as, "Which clients are most likely to respond to my next promotional mailing, and why?"

*Data Mining* derives its name from the similarities between searching for valuable business information in a large database - for example, finding linked products in gigabytes of store scanner data - and mining a mountain for a vein of valuable ore. Both processes require either sifting through an immense amount of material, or intelligently probing it to find exactly where the value resides. Given databases of sufficient size and quality, *Data Mining* technology can generate new business opportunities by providing these capabilities:

- Automated prediction of trends and behaviours. *Data Mining* automates the process of finding predictive information in large databases. Questions that traditionally required extensive hands-on analysis can now be answered quickly directly from the data. A typical example of a predictive problem is targeted marketing. *Data Mining* uses data on past promotional mailings to identify the targets most likely to maximize return on investment in future mailings. Other predictive problems include forecasting bankruptcy and other forms of default, and identifying segments of a population likely to respond similarly to given events.
- Automated discovery of previously unknown patterns. *Data Mining* tools sweep through databases and identify previously hidden patterns in one step. An example of pattern discovery is the analysis of retail sales data to identify seemingly unrelated products that are often purchased together. Other pattern discovery problems include detecting fraudulent credit card transactions and identifying anomalous data that could represent data entry keying errors.

*Data Mining* techniques can yield the benefits of automation on existing software and hardware platforms, and can be implemented on new systems as existing platforms are upgraded and new products developed. When *Data Mining* tools are implemented on

high performance parallel processing systems, they can analyse massive databases in minutes. Faster processing means that users can automatically experiment with more models to understand complex data. High speed makes it practical for users to analyse huge quantities of data. Larger databases, in turn, yield improved predictions.

Databases can be larger in both depth and breadth:

- *more columns*: analysts must often limit the number of variables they examine when doing hands-on analysis due to time constraints. Yet variables that are discarded because they seem unimportant may carry information about unknown patterns. High performance *Data Mining* allows users to explore the full depth of a database, without preselecting a subset of variables;
- *more rows*: larger samples yield lower estimation errors and variance, and allow users to make inferences about small but important segments of a population.

The most commonly used techniques in *Data Mining* are:

- *artificial neural networks*: non-linear predictive models that learn through training and resemble biological neural networks in structure;
- *decision trees*: tree-shaped structures that represent sets of decisions. These decisions generate rules for the classification of a dataset. Specific decision tree methods include Classification and Regression Trees (CART) and Chi Square Automatic Interaction Detection (CHAID);
- *genetic algorithms*: optimization techniques that use processes such as genetic combination, mutation, and natural selection in a design based on the concepts of evolution;
- *nearest neighbour method*: a technique that classifies each record in a dataset based on a combination of the classes of the  $k$  record(s) most similar to it in a historical dataset (where  $k \geq 1$ ). Sometimes called the  $k$ -nearest neighbor technique;
- *rule induction*: the extraction of useful if-then rules from data based on statistical significance.

Many of these technologies have been in use for more than a decade in specialized analysis tools that work with relatively small volumes of data. These capabilities are now evolving to integrate directly with industry-standard data warehouse and OLAP platforms.

Some successful application areas include:

- a pharmaceutical company can analyse its recent sales force activity and their results to improve targeting of high-value physicians and determine which marketing activities will have the greatest impact in the next few months. The data needs to include competitor market activity as well as information about the local health care systems. The results can be distributed to the sales force via a wide-area network that enables the representatives to review the recommendations from the perspective of the key attributes in the decision process. The ongoing, dynamic analysis of the data warehouse allows best practices from throughout the organization to be applied in specific sales situations;



- a credit card company can leverage its vast warehouse of customer transaction data to identify customers most likely to be interested in a new credit product. Using a small mailing test, the attributes of customers with an affinity for the product can be identified. Recent projects have indicated more than a 20-fold decrease in costs for targeted mailing campaigns over conventional approaches;
- a diversified transportation company with a large direct sales force can apply *Data Mining* in order to identify the best prospects for its services. Using *Data Mining* to analyse its own customer experience, this company can build a unique segmentation identifying the attributes of high-value prospects. Applying this segmentation to a general business database such as those provided by Dun and Bradstreet can yield a prioritized list of prospects by region;
- a large goods company can apply *Data Mining* to improve its sales process to retailers. Data from consumer panels, shipments, and competitors' activity can be applied to understand the reasons for brand and store switching. Through this analysis, the manufacturer can select promotional strategies that best reach their target customer segments.

Each of these examples have a clear common ground. They leverage the knowledge about customers implicit in a data warehouse to reduce costs and improve the value of customer relationships. These organizations can now focus their efforts on the most important (profitable) customers and prospects, and design targeted marketing strategies to best reach them.

#### 1.2.4 Digital ecosystem

A *Digital Ecosystem* is inspired to the *Biological Ecosystem*. The *biological ecosystem* is composed by two main elements:

- *species*: there can be multiple species and they need to interact and balance each other (even though some species may play a leading role at times). Members of species are called *Individuals*. Each *Individual* can again be considered as an entire ecosystem;
- *environment*: must supports the needs of its species so they can continue generation after generation.

Boley [12] has individuated four essential aspects of an ecosystem:

- *interaction and engagement*: the inter-species and environment interactions for the social well-being in order to share and defend resources as a unique group against threats from human interference, pollution or natural disasters;
- *balance*: harmony, stability and sustainability within an ecosystem. A single point of failure will not lead to a disaster but may become a contribution to a new balance of welfare for the ecosystem as a whole;
- *domain clustered and loosely coupled species*: loosely coupled species that have similar culture, social habits, interests and objectives can originate a group. Each

specie preserves the environment and it is proactive and responsive for its own benefit. Grouped species are able to live together and support each other for sustainability;

- *self-organization*: each species is independent, self empowered, self prepared, undertakes self defence, self surviving and undertakes self coordination through swarm intelligence.

Boley makes a comparison between the existing network architectures with the *Digital Ecosystem* paradigm and he does not find an existing reference to use. Below, for each network architecture, the contradictions with *Digital Ecosystem*:

- *client-server*: the communications are centralized and act as to to control the environment;
- *peer-to-peer*: at any time, each agent has a well defined role and it can only be a client or a server but not both simultaneously;
- *grid*: it ties partners together for resource sharing but it is not possible to avoid the *Free Riding* [43] problem;
- *web services*: brokers are centralized, service requesters and providers are distributed in a hybrid architecture that does not guarantee trust and QoS.

*Digital Ecosystem* instead is an open community and there is no permanent need for centralized or distributed control or for single-role behaviour. A leadership structure may be formed and dissolved in response to the dynamic needs of the environment. Usually, the *Digital Ecosystem* architecture is implemented like a *Service Oriented Architecture(SOA)*. Ferronato [35] deepens the related problematic regarding this type of approach and he raises a possible theoretical solution to implement *Digital Ecosystem* architecture features. However, *SOA* is not suitable to implement *Digital Ecosystem* because:

- any functional or structural aspect is managed via a central governance entity and the infrastructure is under control and managed via a single department unit (it has a single reference model);
- the functional specifications are planned in advance or in joint meetings among parties or defined by a single central authority and so, it is a hierarchical structure;
- *Web Services Description Language (WSDL)* represents the common technical contract for service invocation that is defined up front and has to be used by all the partners;
- in the *Universal Description Discovery and Integration (UDDI)* there is no separation between the technical and the functional specifications. *UDDI* is conceived as a static catalog of services and so, it is not dynamic;
- if a service finds that the *UDDI* is not available, the requesting service is not able to know the reason of that failure (e.g. if it is a temporary or technical issue);
- the service instances are not resolved at run-time.

### 1.2.5 Domotics

The slang of new technologies is always enriched with new terms as *Home Automation*, *Building Automation*, *Smart Home*. These terms are all synonyms diverted by the *Domotique* neologism. *Domotique* is a word coined in France and it is composed by the fusion of two terms: *Domus* and *Informatique*. The neologism *Domotique* is recovered in English with the term *Domotics* [52] and it represents the technological applications of the information and communication in the domestic World, to offer:

- *comfort*: domotic applications must make more pleasant the space in the house facilitating the management especially for the disabled users (for example, management of audio/video streams, control and supervision of the remote appliances, shine, blinds, gates, etc.);
- *security and safety*: the domotic applications have to guarantee the safety of the user facing situations of emergency (for example, techniques of anti intrusion, survey of fires, escapes of gas, etc.);
- *more independence*: even for people with special needs;
- *energy saving*: the employment of the domotics has to allow a more accurate and efficient management of the energetic consumptions (for example, advanced tools to get information that allow a more equitable distribution of the energetic loads);
- *entertainment*;
- *remote control*;
- *access to external services*,

using:

- user friendly interfaces;
- mobile and wireless technologies;
- integration;
- communication;
- digital networks;
- ...

The domotics includes all the technologies that are dedicated to the integration of electronic devices, appliances, systems of communication and control placed inside a domestic environment.

Domotic view is still in development. The industry and the market has played (and still play) a role of primary importance for the definitive take-off of the domotic technologies. Different industry coalitions have promoted numerous and always more sophisticated middle-wares and standards that are always too poorly interoperable each others. This situation has made the domotics an unexploded bomb.

The worlds of research and industry are waiting for the “boom” that is not still happened. The real employment of the domotic technology is still hindered by the attempt of each industry to impose its standard on the others. The presence of such a vast number of domotic standards announces that there will hardly be a definitive consecration of

one of them. It is as much unlikely that all the coalitions would gather about to realize an unique middle-ware that represents the standard de-facto for the domotics.

Paradoxically, the different and poorly interoperable domotic technologies are so numerous causing an obstacle to the expansion of the market. For the final consumer point of view, the lack of interoperability between the different domotic standards limits his freedom. The consumer will in fact be forced to acquire only the products conforming to a particular system and it could happen in two conditions: (i) the contemporary acquisition of all the needed devices in the building, or (ii) a technical knowledge that allows the user to acquire devices conforming to those he already owns. Both conditions are however difficult to meet because in most cases, the domestic environment is very dynamic: the topology and the removal of its elements change frequently and the devices are acquired in different moments.

Try to think, for example, to the appliances: it is unlikely that all home equipment are purchased at the same time. To better understand this limitation, there is a classical and practical example of the domotic application: lets suppose to have a coffee pot that is able to alight with a radio alarm at a pre-established hour. If the radio alarm and the coffee pot "speak the same language" (have the same standard), every morning will be possible to have a coffee ready when the user wakes up. If one of the two devices must be changed, it will be necessary to replace it with another belonging using the same standard. It is instead desirable to allow the consumer to choose the devices independently from the belonging standard: in this way the consumer doesn't have to know the technical detailed of his system but he is however able to exploit all its possible advantages.

### 1.2.6 Internet Of Things

The *Internet Of Things (IoT)* is a scenario in which objects, animals or people are provided with unique identifiers and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction. *IoT* has evolved from the convergence of wireless technologies, micro-electromechanical systems and the Internet.

*IPv6's* huge increase in address space is an important factor in the development of the *Internet Of Things*. The transition to *IPv6* is important not only because the 4.3 billion *IPv4* addresses are running out, but also because the proliferation of Internet-connected devices is creating a new environment of information. Every device that connects to the Internet requires an IP address, and it has been predicted that by 2020 there will be 50 billion Internet-enabled devices in the world. To put that number in perspective, that equates to more than six connected devices per person, based on an expected global population of 7.6 billion people. Thus the move to *IPv6* is necessary as it provides an almost unimaginable number of IP addresses — 18 quintillion blocks of 18 quintillion possible addresses.

A thing, in the *Internet Of Things*, can be a person with a heart monitor implant, a farm animal with a biochip transponder, an automobile that has built-in sensors to alert

the driver when the tire pressure is low – or any other natural or man-made object to which can be assigned an *IP address* and provided with the ability to transfer data over a network. So far, the *Internet Of Things* has been most closely associated with machine-to-machine communication in manufacturing and power, oil and gas utilities. Products built with machine-to-machine communication capabilities are often referred to as being smart. (See: smart label, smart meter, smart grid sensor).

### 1.2.7 Semantic web and ontologies

The *Semantic Web* [44] is an Internet space that includes documents (or portions of them) describing the explicit relations among things, and containing semantic information intended for automated processing by machines. Not only Web pages, but databases, programs, sensors and even household appliances will be able to present data, multimedia and status information in forms that powerful computing agents can use to search, filter and prepare information in new and exciting ways. In the *Semantic Web* an *Ontology* there is a partial conceptualization of a given knowledge domain, shared by a community of users, that has been defined in a formal, machine-processable language for the explicit purpose of sharing semantic information across automated systems.

In reality many definitions exists for the *Ontology* concept. The most cited and prevalent is attributed to Tom Gruber [38] who defines it as "*an explicit specification of a conceptualization*". This definition derives from the *Declarative Knowledge* of the *Artificial Intelligence* field where the first-order predicate calculus are used to describe models of the world because natural languages are too ambiguous for machine interpretation. The *Conceptualisation* is a simplified view of the world or of the *Domain of Interest* we are representing.

*Ontologies* was born from the philosophical science that studies the nature of being. This discipline was introduced by Aristotle and it attempts to address the question "*What is being?*" and "*What characteristics do all beings have in common?*" [40].

In computer and information sciences, "*an ontology is a designed artifact consisting of a specific shared vocabulary used to describe entities in some domain of interest, as well as a set of assumptions about the intended meaning of the terms in the vocabulary*" [40]. Chandrasekaran [19] explains that "*an ontology is a representation vocabulary, often specialized to some domain or subject matter*". More precisely, it is not the vocabulary as such that qualifies a ontology, but the conceptualizations that terms in the vocabulary are intended to capture. Translating the terms from one language to another does not change the ontology conceptually. For example, we can refer to the word *mouse*. *Mouse* can be the animal or the input computer device. The *mouse* word can therefore represent two different concepts and it is important to know which one is referred to in the ontology. In an ontology, instead, it is not important the language used for the concept that is described. It is possible to call the input device as "*mouse*" in English, "*muis*" in Afrikaans or "*Maus*" in German. The important is that names refer all to the same concept.

*Ontologies* are usually combined with a *Semantic Reasoner*. A *Semantic Reasoner* is a piece of software able to infer logical consequences from a set of asserted facts or

axioms introduced in the *Ontology*. There are many reasoners that take advantage of first-order predicate logic to perform reasoning. The benefits obtained using *Ontology* are described by Happel [42] who underlines all the advantages related to the use of *Ontologies* in software engineering development process.

*Ontologies* are often used to implement semantic interoperability. Some technical methods to realize interoperability are explained by Lopez [85] and Castano [17].

### Development of Ontologies

The development of an ontology can be classified according to the specificity of its description:

- *top-level*: describes very general concepts or common sense knowledge in a coherent, consistent and independent domain way;
- *domain*: describes the categories of a certain discipline and it is applied to a specific application domain (e.g. medicine, chemistry). It is not used directly to build systems or information stores (knowledge bases) but instead, it helps to establish cooperative working agreement terms and meaning of a domain to be understandable by team members belonging to different cultural backgrounds;
- *task*: defines the topics of a field. A field can be a discipline, a industry or any area of a company that unifies many application domains (e.g. diagnostics, sales). A discipline can require different domain *Ontologies*;
- *application*: describes the knowledge specialization of a domain or task.

#### *Ontology development steps*

The steps that are typically required for the development of an ontology are:

- *acquire the knowledge of the domain*: in this first phase, the collaboration with domain experts is useful to try to gather as much information as possible about the domain of interest and to understand the terms used, to formally be able to describe the entities in a consistent manner. The purpose of this step is to answer at the following questions:
  - which domain ontology will be covered?
  - which is the purpose of the ontology?
  - what types of questions is it possible to provide an answer to using the information expressed by the ontology?
  - who will use and who will be responsible for the maintenance of the ontology?
- *consider the re-use of existing resources*: once the domain is characterized, it can be useful to check the existence of resources that can be reused. In fact, the idea to complete and to extend existing resources, such as glossaries and dictionaries of terms and synonyms, documents and so on, is an advantage in terms of development and time of creation.

- *plan ontology*: design the conceptual structure of the domain, identifying the main concepts and their properties. Consider the relations between the various concepts and create an abstraction of them, specifying which of these are instances etc. It's useful to answer at the following questions:
  - what are the important terms?
  - what are the properties?To do that, there are three basic steps:
  - developing a flat glossary to formalize each term using a natural language definition and also providing appropriate examples in which the names become objects or actors, and the verbs are transformed into relationships or processes;
  - develop a structured glossary to decompose and / or specialize the terms and identify the attributes of the concepts;
  - identify all of the conceptual relationships between objects.
- *organize and integrate the ontology*: adding concepts, relations and entities to reach the necessary level of details to reach the objectives of the needed ontology;
- *check your work*: the final developed ontology, must be analyzed to detect any logical and semantic inconsistency among its elements. This phase often facilitates the automatic classification process that leads to highlight new concepts on the basis of the properties of entities and relations among classes;
- *deliver the ontology*: it is necessary that domain experts check the correctness and validity of the ontology to proceed with the delivery of the product and of all the related documents. An ontology is valid if it has the following features:
  - *completeness*: predict all the distinction keys;
  - *conciseness*: do not make implicit and redundant assumptions;
  - *consistency*: do not contain contradictory definitions;
  - *coherence*: allow the presence of all and only the consistent relations for the definition of concepts;
  - *re-usability / scalability*: extend the ontology without modifying already existing concepts.

However, there is not a uniquely correct way to model an ontological domain because it represents a description of a particular reality and the concepts defined in it reflect only a piece of that world. It is highly unlikely that an ontology may exhaustively contain all information and properties available in the domain that it expresses. It appears evident that the user experience helps in the ontology and documentation processes construction. It is useful to document each step of the development phase and in particular regarding the encountered problems and the proposed solutions, to help users and designers in later changes. The work of modeling is not easy but the use of tools that provide a graphic visualization of the ontology, suggesting even automatically concepts and relationships, avoiding to write the code by hand, can be of paramount importance.

### *Specification languages*

To be useful, *Ontologies* must be expressed in a concrete notation. A *language for ontologies* is a formal language used to build an ontology. To meet the needs of an ontology, the language for ontologies must possess these requirements:

- extend existing Web standards to simplify its use;
- be easy to understand and use;
- be specified in a formal way;
- have an adequate expressive power to describe the domain.

There are different proprietary and standard languages to define ontologies.

A *Metadata* is a *Resource* able to provide information about itself. It should be written in a machine universally readable way. Adding semantics to web content requires the creation of languages and technologies that can extract meaning from information, that can express data and rules for reasoning. The term *Resource* is used since the creation of the web to indicate anything available in Internet through the use of its protocols. The generality of the *Resource* term has encouraged a process of generalization on the methods to access to these resources: from the idea to simply locating a resource (*Uniform Resource Locator - URL*), passing to the idea of being able to identify a resource regardless of its location (*Universal Resource Identifier - URI*) to define semantic through links (*Resource Description Framework - RDF*).

In general, any logic programming language, like *Prolog* [5], can be used to build an *Ontology*. Anyway, other languages were created and specialized on the description of ontology logic like *Loom*, *DAML+OIL* and the *Ontology Web Language (OWL)* standard. Instead, it is not reasonable to realize a semantic structure using relational and *XML* databases. In fact, these kind of databases use a composition and formatting of information bind to capture all the semantic in the application logic. Ontologies, however, are able to provide a specification of domain information by representing a consensual agreement between the concepts and relations. In this way it is possible to characterize the knowledge in the described domain.

### *The RDF language*

The *RDF* is a tool for the coding, exchange and reuse of structured *Metadatas*. *RDF* allows interoperability among applications that exchange information using Web permitting automatic elaboration.

The *RDF Data Model* is very easy and it is based on three kind of objects that are all always referred by *URI*:

- *resources*: everything described by an *RDF Expression* is a *Resource*. A *Resource* can be a web page or a part of it, an *XML* element inside a source document. A *Resource* can be a collection of web pages or an object that it is not directly accessible through web (a book, a picture and so on);



- *properties*: a *Property* is a specific aspect, a characteristic, an attribute or a relation used to describe a *Resource*. Every *Property* has a specific meaning and it defines admissible values;
- *statements*: a *Statement* is a *Resource* with a *Property*, a *Name* and a *Value*.

The basic syntax to define a *RDF Document* provides the conceptual links by defining *Predicates* (or properties) that relate a subject to an object. A statement is a triple of type: Subject - Predicate - Object where the *Subject* is a *Resource*, the *Predicate* is a *Property* and the object is a *Value*.

The cases in which *RDF* can be applied to bring benefits are:

- to describe the contents of a Web site, a page or a digital library;
- to implement intelligent software agents for the exchange of knowledge and best use of Web resources;
- to classify the content to apply selection criteria;
- to describe a set of pages that represent a single logical document;
- with the mechanism of the digital signature, to contribute to the creation of the Web of Trust, for applications in electronic commerce, cooperation, etc.

The *RDF*, therefore, does not describe the semantics but provides a common basis to express itself, allowing to define the semantics of *XML Tags*. *RDF* is formed by of two components:

- *RDF Model and Syntax* that define the *RDF Data Model* and its *XML Encoding*;
- *RDF Schema* that allows to define specific vocabularies for *Metadata*.

### *The OWL language*

Whilst *RDF* provides much more subtle and powerful syntactic details, it is not sufficient to implement features in the semantic web applications. In order to get a full semantic qualification there is the need to add special constraints to the *RDF Syntax*. This purpose is reached by the *Ontology Web Language (OWL)*, a language of the *Description Logic* family and expressed by using *XML* syntax. *OWL* is developed as the next step of *RDF* and *RDFS* as well as previous ontology languages such as *Ontology Inference Layer (OIL)* and *DAML*. *OWL* is the current standard for ontologies in Web environments. The purpose of *OWL* is not only to allow the attribution of meaning to resources, but also to make such meanings computable using automatic mechanisms to evaluate inferences about these meanings. According to a well-established information technology practice in Object Oriented programming languages, the resources within the *RDF* and *OWL*, are classified as *Classes* (usually to abstract and conceptualize objects) and *Instances* (to represent concrete objects). *OWL* has been released in three different versions that differ in complexity and expressiveness:

- *OWL-Lite* is the syntactically simpler version, through which it is possible to define class hierarchies using low complexity constraints;

- *OWL-DL* is the intermediate version and it is defined as *DL* for its correspondence with the *Descriptive Logic* language. A *descriptive language* is simpler than a *first-order language* because it contains only atomic *Concepts*, *Roles* and *Names* of objects. A *Concept* is an atomic common name as *father*, *wife*, etc.; a *Role* is a binary relation and an object *Name* is just a single object. *OWL-DL* has a high expressive power and maintains computational completeness (all conclusions are computable) and decidability (all computations end in a finite time);
- *OWL-Full* provides maximum expressiveness without any guarantee regarding completeness and decidability.

Each version of the language includes and extends the previous one, thus *OWL-Lite* ontology is always valid in *OWL-DL*, *OWL-DL* ontology is always valid in *OWL-Full*.

The main components of *OWL* ontology are:

- *individuals*: represent objects in the domain of interest;
- *properties*: binary relations (i.e. linking two objects at a time) between individuals,
- *classes*: groups of individuals.

### Ontology editors

The development process often involves solutions using numerous ontologies that can be linked from external or internal sources. Linked ontologies may progress through a series of versions and became necessary to keep track of them. It can be convenient to use tools that can help to map, to link, to compare, to merge, to convert and to validate them. When starting out an ontology project, it is reasonable to find a suitable ontology software editor.

#### *Protégé*

An important software used for the ontology creation is *Protégé* [83]. *Protégé* is an *Open Source OWL* editing tool developed by the *Stanford Center for Biomedical Informatics Research* at *Stanford University School of Medicine* that includes a growing user community and a suite of tools to create models that cover several domains: from the medical (to model the spread of disease) to the military (for the management of nuclear power plants) fields. *Protégé* helps domain experts to build applications for the information management.

The developers of ontologies have access to relevant information in a simple and fast way and they can use tools to directly manipulate, and to easily and quickly navigate among ontologies and their class hierarchies.

The *Protégé* platform supports two main ways of modelling ontologies:

- *Protégé-Frames editor*: allows users to build and populate ontologies that are based on *frame*, in accordance with the *Open Knowledge Base Connectivity (OKBC)* protocol. In this model, an ontology is composed by a set of classes organized in a hierarchy that represents concepts. The classes are characterized by properties (*Slots*) and *Relations*;

- *Protégé-OWL editor* allows users to create ontologies for the *Semantic Web*. OWL ontology may include descriptions of classes, properties and their instances. Besides the presence of a simple interface *Protégé* provides:
  - support for classes and class hierarchies;
  - a variety of template slots ready to be used;
  - specifications of the attributes of the slots, which include allowable values, restrictions on the cardinality, defaults;
  - *metaclasses* (classes to handle the classes of domains) and hierarchies of *meta-classes*.

Two other features that distinguish *Protégé* by many other development environments are the scalability and the extensibility. The system is modular and its architecture is based on components that simplify the addition of new functionalities through the *Protégé Plugin Library* that collects plug-ins created by developers around the world. The plug-ins are used for example to provide visualization capabilities, advanced control versions, and so on. An example is *OntoViz* that displays an ontology as a graph using an open source library optimized for graphical display: classes and instances are represented as nodes and relations are displayed as strings. Both nodes and arcs are classified and arranged in such a way as to minimize overlaps at the expense of the size of the graph. Another example is for the 3D visualization of ontologies: *Ontosphere3d*, a software developed by *E-Lite* group of *Politecnico di Torino*.

### 1.2.8 Machine Learning

*Machine Learning* is sub set of *Artificial Intelligence* and it is a study of systems that can learn from data. A *Machine Learning* system could be trained. Core of *Machine Learning* deals with representation and generalization.

Machine learning is a "Field of study that gives computers the ability to learn without being explicitly programmed". A core objective of a learner is to generalize from its experience. Generalization in this context is the ability of a learning machine to perform accurately on new, unseen examples/tasks after having experienced a learning data set.

The goal of machine learning is to program computers to use example data or past experience to solve a given problem. Many successful applications of machine learning exist already, including systems that analyse past sales data to predict customers' behaviour, optimize robot behaviour so that a task can be completed using minimum resources, and extract knowledge from bioinformatics data.

The main differences between *Machine Learning* and *Data Mining* are:

- *Machine Learning* focuses on prediction, based on known properties learned from the training data;
- *Data Mining* focuses on the discovery of (previously) unknown properties in the data (this is the analysis step of *Knowledge Discovery* in Databases);
- *Data Mining* uses many *Machine Learning* methods, but often with slightly different goals;

- *Machine Learning* also used *Data Mining* methods as "unsupervised learning" to improve learner accuracy;

Typical *Machine Learning* algorithm types are:

- Supervised learning (labelled);
- Unsupervised learning (unlabelled);
- Semi-supervised learning;
- Transduction (reasoning from observed);
- Learning to learn (multi-task learning);
- Reinforcement learning;
- Developmental learning (imitation);

Typical applications for *Machine Learning* are:

- Computer vision (object recognition)
- Natural language processing
- Syntactic pattern recognition
- Search engines
- Medical diagnosis
- Detecting credit card fraud
- Stock market analysis
- Speech and handwriting recognition
- Game playing
- Software engineering
- Adaptive websites
- Computational advertising
- Computational finance

### 1.3 Thesis Structure

The document is organized as follows: the "Introduction" chapter introduces the concepts used in this work. Then, in the "State of the Art" chapter, are mentioned some of the most important research projects that make use of the innovative technologies considered to design and implement the presented architecture. In the descriptions of the projects it is also outlined the current state of the art of the research in the related field. The "Architecture" chapter contains an overview about the architecture and the design choices used to implement the software platform. The following "Domotic agent", "User agent" and "Intelligent agent" chapters deepen the different components of the architecture explaining how they work and how they interact with the other parts of the software. Finally, the "Test and Verification" chapter describes the evaluation of the software and then, "Conclusions and future works".

## State of the art

### 2.1 Ambient Assisted Living

#### 2.1.1 HERA

The *HERA* [78] (*Home sERVICES for specialized elderly Assisted living*) project aims to provide a platform with cost-effective specialized assisted living services for the elderlies suffering of mild Alzheimer with identified risk factors. The aim is to significantly improve the quality of their home life and, at the same time, reinforce their social networking. The *HERA* platform will provide three main categories of services:

- *cognitive and physical reinforcement services*: these services will be a supplement of non-drug therapeutic interventions provided to the patient by specialized Alzheimer care centers;
- *patient specific home care services*: this service category will include social reinforcement services, reality orientation support services and services capable to monitor several Alzheimer related risk factors;
- *general home care services for elderly*: this service category will include medication reminder services, information services as well as alarm services in cases of abnormal health conditions.

#### 2.1.2 HOMEdotOLD

The *HOMEdotOLD* project [68] (*HOME services aDvancing the sOcial inTeractiOn of eLDerly people*) aims to provide a TV-based platform with cost-effective services that will be delivered in a highly personalized and intuitive way and will advance the social interaction of elderlies, aiming at improving the quality and joy of their home life, bridging distances and reinforcing social voluntaries and activation, thus preventing isolation and loneliness.

The project main objectives are to:

- provide the appropriate platform based on *INHOME* and *Net TV* technologies for supporting the services described above advancing the social interaction of the elderlies;

- provide services allowing the elderly to stay socially active including the *social working* and the *personalized news headlines* services;
- provide services for bridging distances and supporting elderly's existing roles, including the *video-conference*, the *remote dining*, the *photos*, *videos*, *experience sharing* and the *intelligent calendar* services;
- install the *HOMEdotOLD* pre-product prototype and perform trials at pilot sites by involving real elderlies;
- evaluate and demonstrate the commercial feasibility and the business potential of the *HOMEdotOLD* services by drafting a realistic business and exploitation plan.

### 2.1.3 I2HOME

*I2HOME* project [4] (*Intuitive interaction for everyone with home appliances based on industry standards*) will address the problem of living an independent life and realizing full potential with an approach based on existing and evolving industry standards. The project will focus on the use of home appliances for elderlies and persons with cognitive disabilities. At the same time the project will take care that the developed and standardized access strategies will be applicable to domains beyond the home. *I2HOME* is a *STREP* project launched on September 1st, 2006 and has an estimated duration of 36 months.

In *I2HOME*, participants will build upon a new series of industry standards (ANSI/INCITS 389ff) for interfacing networked appliances by means of a *Universal Remote Console (URC)*. The participants will use an architecture with a *Universal Control Hub (UCH)* as core component that communicates to networked (off-the-shelf) home appliances and consumer electronic devices through industry networking protocols. The user interfaces will be designed according to the results of a broad requirements analysis and will include multi-modal communication and activity management.

### 2.1.4 MPower

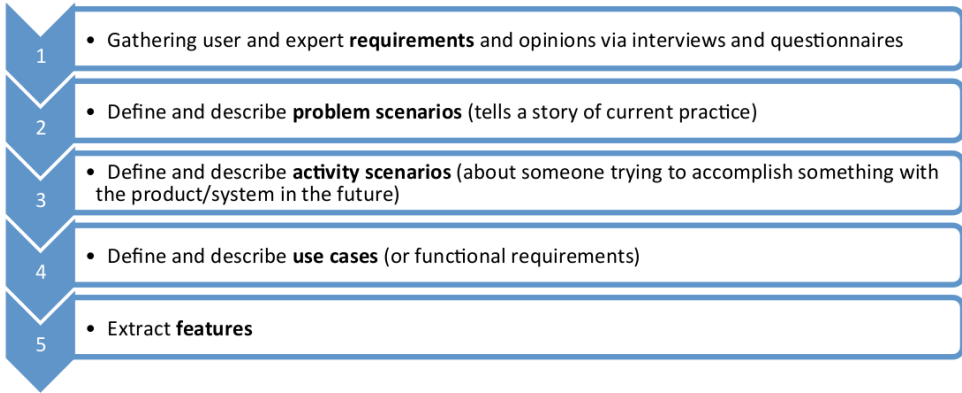
The FP6-STREP *MPower* project is aimed at defining and implementing an open platform as a suite of independent building blocks to simplify and speed up the task of developing and deploying services for persons with cognitive disabilities and elderlies. The platform-related goals of the project were to support:

- the integration of smart house and sensor technology;
- interoperability between profession and institution specific systems;
- secure and safe information management;
- including both social and medical information;
- mobile users which often change context and tools.

### Use case methodology

The target groups in the *MPower* were elderlies, people with dementia/cognitive decline and relatives.

In general *MPower* follows an iterative process in identifying scenarios, Use Cases and features, as explained in Figure 2.1.



**Figure 2.1.** MPower use case specifications

In *MPower* project, two types of scenarios were used:

- *Problem Scenarios*: tells a story of current practice - it is developed according to the gathered user and expert requirements;
- *Activity Scenario*: tells a story about someone trying to accomplish something with the product/system in the future.

Overall eighteen problem scenarios have been developed and several activity scenarios were elaborated from them. Each of these activity scenarios usually have more than one actor, which resulted in a lot of Use Cases. Several of the activity scenarios were overlapping therefore those scenarios were identified and more general Use Cases were created.

## Requirement Analysis

Requirements analysis in *MPower* project has been based on group interviews and questionnaires prepared for the target groups in the partner countries consisting of elderlies in a senior's home in the Netherlands, relatives/family carers of people with dementia in Austria and Norway, and experts on elderlies and dementia in Austria, Poland and Norway.

The prioritization of the functional requirements was done based on the following criteria:

- it should reflect a challenge for family carers, for professional care givers, for service providers and for society;
- it should improve everyday living for elderlies and people with cognitive impairments and dementia;

- it should possibly lead to cost effectiveness regarding safety and security (safety value);
- it should cover a future need for society and provide improvements of care services.

### 2.1.5 OASIS

*OASIS (Open architecture for Accessible Services Integration and Standardization)* is an Integrated Project with the scope to revolutionize the interoperability, the quality and the usability of services for all daily activities of elderlies. More specifically, *OASIS* project intends to utilize ICT and other key technologies in order to provide holistic services to elderlies to support their physical and psychological independence, to stimulate their social or psychological engagement and to foster their emotional well being. *OASIS* is the acronym of a Large Scale Integrated Project co-financed by the European Commission (7th Framework Programme, ICT and Aging - Grant Agreement No: 215754). The full project name is: Open architecture for Accessible Services Integration and Standardization. It started on January 1st, 2008 and it lasted four years. The *OASIS* Consortium was composed of 33 Partners from 11 countries. Large Industries, SMEs, Universities, Research Centers, Non-Profit Organizations, Public Organizations and Healthcare Centers are all represented.

### Use case methodology

Benchmarking was the first step for Use Case definition. A thorough investigation was carried out on the relevant technological aids, systems and services able to support AAL environment, as well as the lack or existence of relevant ontologies in the different domains. As a result, an online database has been implemented to facilitate the compilation of information and analysis of all the identified Products, Services and Research Projects in the area of Independent Living Applications, Autonomous Mobility and Smart Workplaces area for elderlies. The aim of benchmarking was to define the context framework in which Use Cases of *OASIS* would be specified.

The first step of benchmarking (Figure 2.2) was to create a template in order to determine the most relevant information needed. Based on this template, the database was defined and the gathering of information was made.

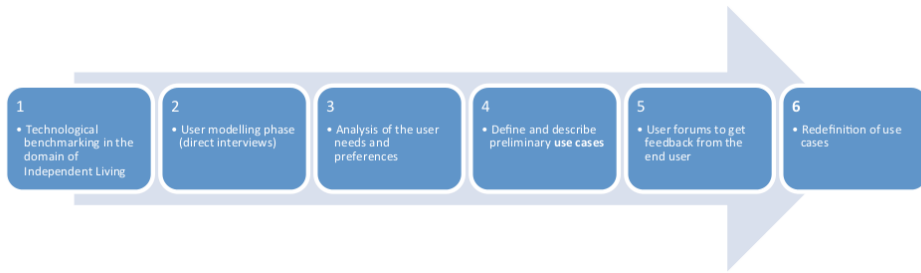
Use Cases are generated using a goal-oriented methodology: examining all the actor's goals that the system satisfies yields the functional requirements. Use Cases were the goals that were made up of scenarios.

After the first definition of the Use Cases, these were validated in different forums to get feedback to the user and the Use Cases were redefined. The objective of this process is to achieve the participation of the user in service definition phase.

### Requirement Analysis

Different types of questionnaires were devised for collecting user needs and requirements based on the different sub-project topics and areas, e.g. the Questionnaire on In-





**Figure 2.2.** OASIS use case specifications

dependent Living Applications and the Questionnaire on Autonomous Mobility and Smart Workplaces applications.

For each one of the aforementioned questionnaire types, two surveys conducted; one for elderlies and one for caregivers in five European countries (Bulgaria, Germany, Italy, Romania and Spain) and related to the approach of elderlies towards the ageing.

After analyzing the results from the questionnaires, the following domains were identified: Designing for the elderly with adaptability to the characteristics of the elderlies as the key requirement in order to improve technology acceptance for this sector of the population, Personalization in interaction of disabled and elderlies with ICTs, provision of content and information search, ICT in the ageing process with Reassurance and Reminding, Stimulation and Enabling, Belonging and Participation, Safety and Protection as the four clusters of needs, Consumer behaviour using a “gerontographic model”.

The set of the requirements were then extracted from this gathered data by analyzing the statistics and interrelationships and comparing the needs with the achievements of the AAL research so far. As a result, 141 requirements in 18 categories were identified.

### 2.1.6 PERSONA

The *PERSONA* project [80] aims to develop a scalable open standard technological platform and building a broad range of AAL services for the development of sustainable and affordable solutions for the social inclusion and independent living of Senior Citizens. One of the main challenges the project has faced is the design of these solutions in a way that they meet real user needs that they and are psychologically pleasant and easy to use. In order to do this, the project has defined an activity line of work devoted specifically to design User Experience. The goal of this activity line has been to assure the involvement of end-users and stakeholders in the process of defining, developing and validating AAL Services in such a way that these services will provide a total end-user experience from the start to the end, having them embedded in people’s daily context of life in, around, and out of home. These kind of services support people’s exploration of their own boundaries in relation to their social needs, wish for autonomy, security and mobility.

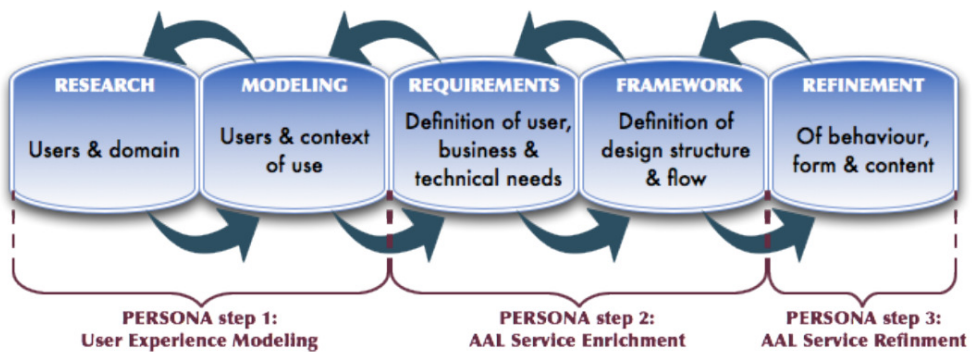
## Use case methodology

The User Experience approach has been defined as an iterative process combining trends research with user experience methodologies with the aim of enabling continuous end-user insights and feedback along the project life-cycle.

The User Experience methodology followed has adopted a human-centered approach according to the principles referenced in *ISO 9241-210*:

- design is based upon an explicit understanding of users, tasks and environments;
- users are involved throughout design and development;
- design is driven and refined by user-centred evaluation;
- process is iterative;
- design addresses the whole user experience;
- design team includes multidisciplinary skills and perspectives.

In order to comply with these principles, the work in UX design has applied the *Goal Oriented Design* methodology proposed by Alan Cooper [30]. The process and links of *PERSONA* work is depicted in the following figure (Figure 2.3):



**Figure 2.3.** Use Case design in the Persona project

The first initial categorization of users, domains and needs has been done based on the definition of the four AAL service domains: 1) *social integration*, 2) *performance of daily activities*, 3) *living safe and protected*, and 4) *mobility*.

During the first step of design, user social inclusion, independence, security and mobility needs have to be identified and analyzed. After that, AAL service scenarios have been specified and they were derived from the analysis made in advance on the services of high potential impact for independent living of senior citizens, the partners experience, the user context delivered by the previous work, and the close collaboration with the pilot sites from an end user point of view. Main focus has been put on describing the user interaction with the system that constituted the user experience model.

The second step was the enrichment phase aiming at improving the service offer by working out the implications of the user experience model resulting in a high-level of specifications of service directions.

The enrichment process consisted in three steps:

1. improve value creation:
  - improve storyline based on end user's feedback;
  - specify the drivers and core values behind the service considering the business aspects;
  - review stakeholders involved in the service by specifying the business role and information flow;
  - review system capability considering feedback of technical experts;
2. subdivide scenarios into functionalities;
  - specify single functionalities drivers based on user needs;
  - list tasks (Use Cases) needed to enable functionality;
3. define interaction paradigms for service access and use;
  - determine common Use Cases;
  - define key touch points between actor and system;
  - specify related assumptions, pre and post conditions;

The evaluation process of the second step consisted of a technical, business and user experience evaluation. Technical analysis was done in order to determine the completeness of the scenarios already described and the level of feasibility according to the technologies to be developed; business analysis was done in order to determine the viability of the scenarios and the complexity to develop a business model around each of them; and user experience analysis was done to determine if user needs were appropriately addressed and proposed services were desirable from the user perspective as well as from the public and private welfare system. Based on the results of this evaluation, scenarios for building mock-ups were selected.

The third step was the refinement phase which objective was to reach a solid service specification by refining the services through iterative stakeholders input loops. Based on the previous work mock-ups were developed and presented to end users and stakeholders with the aim of gathering qualitative insights from the stakeholder perspectives. These qualitative insights supported:

- the process of prioritizing the most attractive services, by providing guidance and ensuring comparability (as much as feasible);
- the further development of services, by validating the concepts and identifying improvement points and strengths.

The analysis of the results was used for improvement of Use Cases and to provide the guidelines for Service specifications. The outcome was delivered to the developer teams in charge of implementing the AAL Services.

### Requirement Analysis

The requirements analysis process in *PERSONA* started in two parallel threads, one with regard to the platform and the other with regard to user requirements:

- *platform-related thread*: the collection of requirements was based on investigation of scientific literature and existing solutions and studies. These investigations were done in the context of a series of state-of-the-art studies in the fields of architecture and middle-ware, context-awareness, service infrastructure, multi-modality and multi-media integration, and privacy-awareness and trust. An interesting aspect here was that *PERSONA*, similar to *universAAL*, had an explicit focus on the creation of an AAL platform and hence tried to reuse existing achievements in the field as much as possible. For this purpose, the platform-related requirements were simultaneously looked at as a set of criteria for evaluating existing solutions.
- *user-related thread*: the requirements analysis started with the creation of an overall user experience model based on the specifications of users' needs in the application domains of social inclusion, assistance in daily activities, safety and security, and mobility, while considering the socio-dynamic factors that influence and shape the acceptance of this kind of services. For determining the socio-dynamic factors, an analysis of the current state of AAL services and technologies was performed focusing on the differences between European countries in general as well as in the pilot sites. The analysis included the following dimensions: socio-economic factors, socio-cultural factors, services and technologies, legal and regulatory environment and ethical aspects.

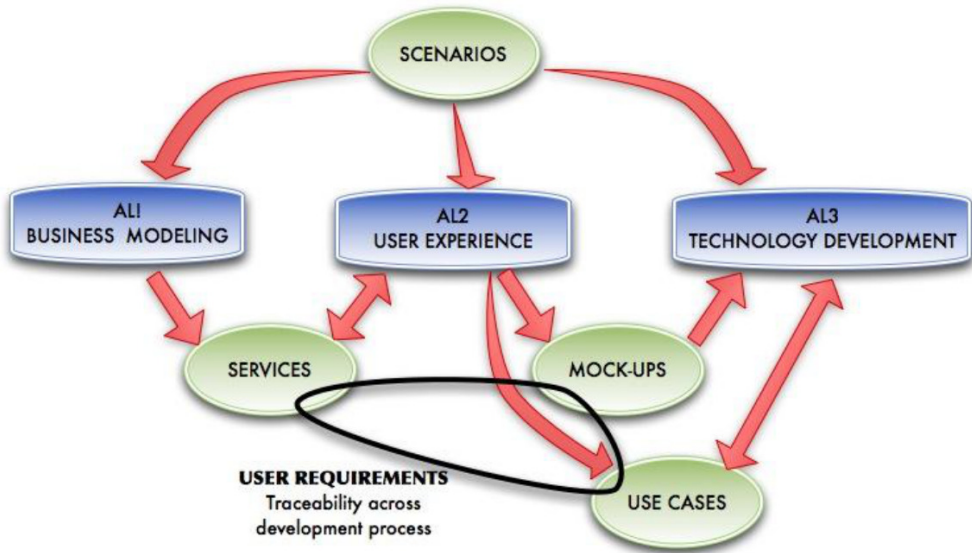
The result of the work performed in this first phase was a set of high-level user requirements based on story-lines where an elderly had solved a specific need by means of *PERSONA* solutions. These scenarios were used to devise the services from a business perspective and the functionalities from a user experience perspective, while using them to derive additional technical requirements.

A first prioritization of scenarios was done based on their added value and usability to the end user, and based on its technical feasibility and system capabilities.

At a second stage, mock-ups created scenarios to gather end-user's feedback. These scenarios were evaluated at the selected pilot sites using individual interviews with elderly as the method to collect qualitative insights together with questionnaires to assess, in a standard and formal way, subjective judgments, attitudes, opinions or feelings about the scenarios presented during the interview.

The results of this assessment were used to refine the scenarios, improving the use cases derived from them as well as the service specifications from the business and technical perspective. A summary of this partially iterative process is shown in Figure (Figure 2.4).

Figure 2.4 also shows that user requirements in *PERSONA* were extracted from a combined analysis of the mock-ups, use cases, and the services. It is worth mentioning



**Figure 2.4.** Persona user requirements analysis process

that users explicitly expressed some requirements during the mock-ups evaluation process or by the domain experts during workshops. This approach has linked two main problems:

- requirements were scattered across different information sources, and
- there was a multiplicity of ways of expressing the requirements depending on their source.

The *VOLERE* [72] methodology was selected due to its capability of solving those problems as it defines a common language for the expression of all requirements, combined in a single document, and it provides the ability to keep track of all requirements and their sources during the development phase.

One of the most important points when defining the user's requirements was the correct definition of the fit criterion. That is, the measurement of the requirement that can be monitored by the system or an evaluator in order to test until which degree the solution matches the requirement. Three types of fit criteria have been defined which are also included in the parameters defined in the pilot sites evaluation methodology:

- capability checked during implementation;
- performance tested in the field;
- benefit evaluated in the field from a long-term perspective.

### 2.1.7 SOPRANO

*SOPRANO* [77] stands for *Service-oriented Programmable Smart Environments for Older Europeans* and it is an Integrated Project in the European Commission's 6th Framework Programme (IST Priority 6th Call on Ambient Assisted Living - AAL).

The project aims to develop an affordable smart ICT-based assisted living service with interfaces which are easy to use for elderlies and relatives in their home environment. The societal trends that *SOPRANO* is responding to are:

- the increase of elder citizens in the population due to demographic change, the scale and type of needs of older citizens which society must plan to meet, the rejection of current ICT-based services by many older citizens, the steady deterioration of non-ICT-based service provision in the Information Society;
- the lack of ICT-based services usable by elder citizens;
- the difficulty of designing ICT-based services that can be used by elder citizens.

*SOPRANO* is developing supportive environments for elderlies based on the concept of *Ambient Assisted Living*, using information and communication technologies (ICTs) to enable elderlies to live independently in their own homes. *SOPRANO* will not only address the problems of old age (e.g., falls, health problems), but will focus on positively enhancing the quality of life of elder people. Focus groups and interviews with elderlies and care providers identified a number of potential opportunities for the development of *SOPRANO*. Social isolation has profound negative outcomes such as loneliness, depression, boredom, social exclusion and disruption of patterns of daily living.

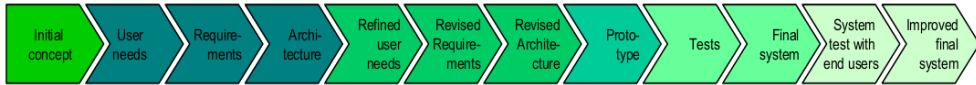
The project foresees an *avatar* (an interactive computer-generated assistant; pictured here) on a TV screen that will be able to interact with the persons in their home using natural language, for example, providing prompts for what exercises to carry out and reminding people to leave their house in a safe and secure manner when they go out (e.g., locking doors, shutting windows).

Social isolation may be alleviated through the more extensive use of video-telephony to link elderlies who live alone with their family and friends. Many of the features of *SOPRANO* will be useful for people with mild dementia: particularly those that help and support people to carry out tasks of daily living. However, extensive research and development is required to ensure that the interaction media (interfaces, avatars, etc.) are appropriate for a diverse range of potential users, such as people with dementia, or those with sensory impairments.

### Use case methodology

*SOPRANO* used the Experience and Application Research (E&AR) methodology to facilitate the active and strong involvement of elderlies throughout the entire R&D process. Participative methods in the area of research and development enable to thoroughly focus on the users when defining the user requirements, iteratively generating design solutions and evaluating those designs in real life settings.

The *SOPRANO* Use Cases development (Figure 2.5) started with composing an extensive list of situations that threaten independence in the life of the elderly. The list comprises circa 80 situations that are of interest to the *SOPRANO* project. These situations were extracted based on extensive literature research.



**Figure 2.5.** SOPRANO use case specifications

The collection of user needs started in a second step with the help of focus groups consisting of elderlies and of formal and informal carers, in four EU countries. Main goal was to elicit user requirements in terms of key challenges that threaten independence together with initial ideas for possible technological solutions.

Use Cases were then further explored in a second stage of user's involvement: the multilevel prototyping. In a first cycle multimedia mock-ups and theatre presentations were used to help users visualize technologies prior to prototyping. Feedback from users was used to refine Use Cases and Requirements. Subsequently also the system architecture was revised and *SOPRANO* component prototypes were developed. After that, prototyping focused on testing usability of the different *SOPRANO* components and resulted in concrete design refinements of certain aspects of the system. Based on this, the final system was developed and it will again be tested. Results from this interaction phase will be used to further improve the system as a whole.

## Requirement Analysis

Based on the Description of Work and further literature analysis, about 80 core user's needs were collected in a short XML-like format. The user's needs focused on situations of concern and situations of assistance highly centred on elderlies living in their own homes. This collection of user's needs served as input to the requirements analysis, which included the first interaction with potential end- user. Within focus groups in four EU countries discussions were held with elderlies, formal and informal carers. Within these discussions relevant needs and situations were identified. The user's needs served as initial guideline in those discussions.

Based on these discussions a set of key challenges to independence and initial ideas for possible technological solutions were derived. Ideas for improving services were developed in response to each challenge to independence, taking into account not only features of technology seen as desirable by focus group participants but also their fears or rejection of other features of technology and of action by service providers seen as intrusive or unnecessary. These key challenges (over 22) acted as the basis for further scenario development.

In a next step, 11 scenarios (in *SOPRANO* also called use cases) were developed that effectively helped to capture the functional requirements describing interactions between users and the system.

In a final step of the first iteration, a list of ca 100 requirements capturing the essence of these scenarios were extracted. These requirements and scenarios then served as basis to further develop the *SOPRANO* architecture and the different *SOPRANO* components. Use cases and requirements were then further explored in subsequent iterations with heavy end user involvement.

The *SOPRANO* project has focused mainly on the requirements of end-users, namely elderlies, formal and informal carers. Therefore most of the requirements are captured in the category of functional requirements. These requirements describe functionalities of certain applications and services from a user's perspective and have been categorized into the groups Reminders, Alerts and messages, User interfaces, Locators, Home management and security, System administration, AP safety and living habits, and General functional requirements.

To capture the refinement according to input from the first and second user cycles, the requirements were collected in a tabular format, e.g. with one column for the original requirement as derived from the first version of the UC, and an additional column for refinements that have to be applied to the original requirement to be compliant with user input from the first and second user cycles.

A far shorter list of non-functional requirements (about 14) mostly captures security concerns and emphasizes *SOPRANO*'s focus on the home of the assisted person. This list has been extended during the process of the project mostly driven by input from technicians but also user's input. In general, due to the strong user's involvement an overly formal approach to requirements and use case specification (e.g. UML-diagrams or enterprise architecture) has been avoided. Exchange and specification formats for UC and requirements were mostly based on MS Word and Excel documents.

### 2.1.8 *universAAL*

*universAAL* [41] aims to reduce user's barriers adoption promoting the development and widespread uptake of innovative AAL solutions. It will benefit end-users (i.e. elderlies and people with disabilities, their carers and family members) by making new solutions affordable, simple to configure and to personalize. It will benefit solution providers by making it easier and cheaper to create innovative new AAL services or adapt to existing ones using a compositional approach based on existing components, services and external systems. The result should be as simple for users to download and setup AAL services as it is to download and install software applications on a modern operating system. *universAAL* establish a store providing plug-and-play AAL applications and services that support multiple execution platforms and can be deployed to various devices and users.

*universAAL* aims to produce an open platform that provides a standardized approach making it technically feasible and economically viable to develop AAL solutions. The platform will be produced by a mixture of new development and consolidation of state of the

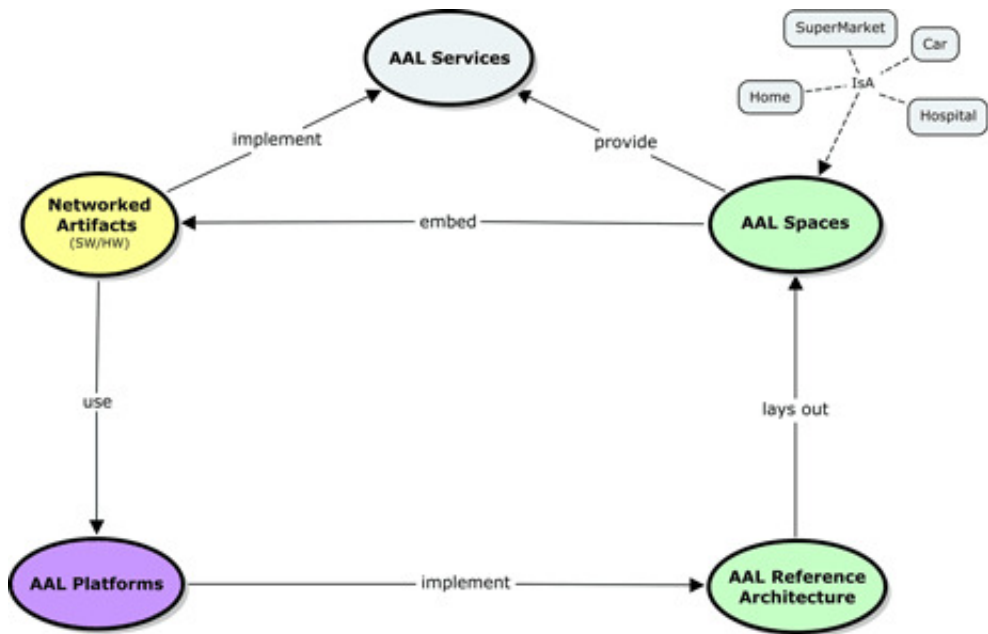


art results from existing initiatives. A variety of similar important projects has been funded in this area in recent years, including *PERSONA*, *MPOWER*, *SOPRANO*, and *OASIS*. In order to achieve a high acceptance of the emerging open AAL platform, *universAAL* will consolidate the earlier work by adopting and integrating earlier results where possible and making new developments where needed.

An early result of the analysis and consolidation of different input projects is the *universAAL reference model*. The model specifies the project's view on the core set of domain concepts and the essential interrelationships among them. The reference model is described as a set of conceptual maps and the *Root Concept Map* (Figure 2.6) presents the consolidated understanding of AAL systems in a single picture using the fewest possible set of concepts. AAL systems are all about the provision of AAL Services. The importance of ambient technologies in the provision of such services is highlighted by putting the concept of AAL Spaces (aka Smart Environments) and the underlying technologies (Networked Artefacts) right in this top level map. The AAL Reference Architecture and the compliant AAL Platforms incorporate the engineering challenges beyond single technologies towards modular and interoperable infrastructures. The AAL Reference Architecture identifies the basic building blocks necessary for constructing an AAL Space, such as homes, supermarkets, cars or hospitals. This facilitates the provision of AAL Services with the help of embedded Networked Artefacts that implement (or contribute to the implementation of) those AAL Services. The cooperation between Networked Artefacts distributed in an AAL Space is facilitated by an AAL Platform that implements the previously mentioned reference architecture in order to provide for resource sharing, context-awareness, and personalization.

AAL spaces are smart environments centred on human users. The devices embedded in such environments operate collectively using information and intelligence that is distributed in the infrastructure connecting the devices. AAL Spaces are classified in space profiles, each identifying the typical set of devices used in a specific AAL scenario; the project distinguish between private space profiles, like homes, versus public space profiles, like supermarkets. Space profiles include industrial standards used by devices that currently populate market segments like e-healthcare, home and building automation (eg ISO/IEEE 11073 standards, IEEE 802.15 standards). The definition of space profiles makes it economically viable to develop heterogeneous products that are still interoperable, thus paving the way to the creation of a promising AAL ecosystem.

Another important aspect of AAL spaces is that they may be remotely managed. This is a typical requirement derived from use cases where a person is assisted by formal and informal caregivers (relatives, neighbors, and friends). Remote access and management of a AAL space can be provided only after a design process that involves various professionals. The idea of standardizing the AAL space design according to a specific profile and following a well-defined process (Figure 2.7) is essential to enable the seamless assistance needed by people traversing different environments. In particular, the commissioning of a system is related to the binding of the distributed services that cooperatively guarantee the basic common services characterising every AAL space (eg



**Figure 2.6.** The root concept map of the universAAL reference model

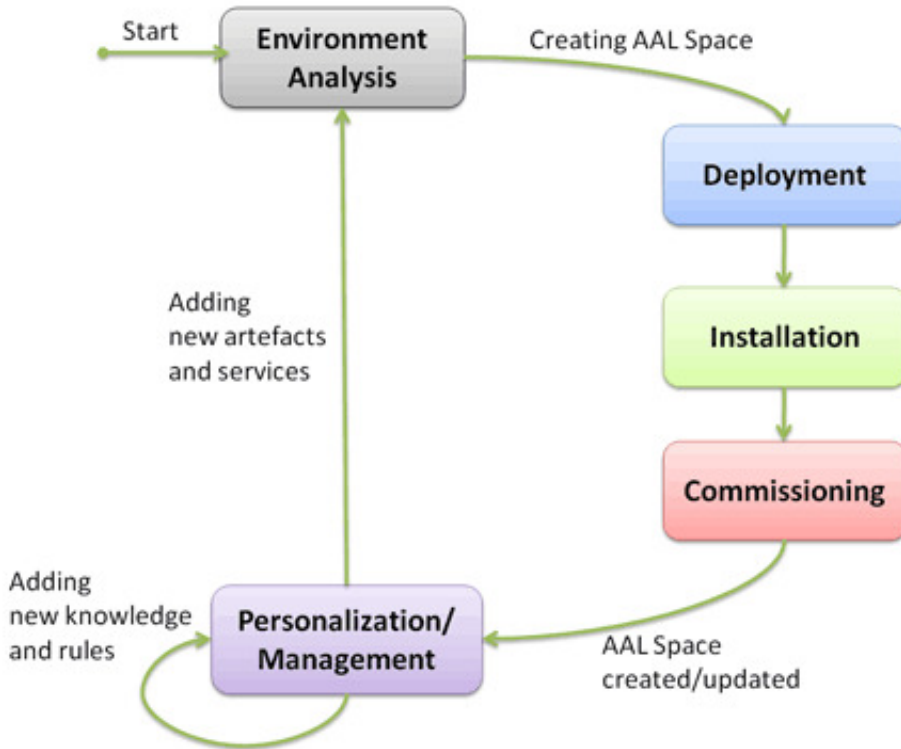
context information provision, user adaptation). In this way, end-users will experience an integrated world, easy to interact with, and based on natural communication where the complexity of the environment is mitigated and hidden by different ICT solutions (platforms) implementing a shared AAL reference architecture.

Work on establishing and running a sustainable community will receive attention right from the start, with promotion of existing results gradually evolving into promotion of the *universAAL* platform, as it develops into one consolidated, validated and standardized European open AAL platform.

### Use case methodology

For the process of alignment of the Use Cases, it was important that the Use Cases collected from the different projects were defined at the same level of detail. Therefore, the Use Case hierarchy in each of the previous projects was identified and an appropriate level for the collection was defined.

The collection step involved identifying similar Use Cases from different projects within a category and consolidating them as a single Use Case. This involved extending certain Use Cases to be more general to cover minor variations of the Use Case in the different projects. Apart from the consolidation of the existing Use Cases from the previous projects, certain Use Cases that were relevant for the *universAAL* specific categories were also added. This last step was not done extensively with the idea that more additions would be made in the future as the project progresses.



**Figure 2.7.** The root concept map of the universAAL reference model

An overview of Use Case categories and related subcategories is depicted in figure 2.8.

### Requirement Analysis

Throughout the whole iterative process, which is described in the next subsection, the *ARCADE architecture development framework* [13] was used. ARCADE defines two models:

- *Requirement Model*: its purpose is to identify, and eventually specify all relevant requirements to the target system where a requirement shall be verified. According to this model, it is mandatory that each requirement is uniquely identifiable and testable for the target system verification. Furthermore, the model recommends to hierarchically decompose requirements to one or more levels;
- *Target System Information Model*: the purpose is to be a supplementary specification to the Requirement model in order to obtain a more complete, easier and understandable specification of the target system interfacing to its environment. The main idea is to include only requirements specification related to the target system interfacing with its environment. This model must include a specification of who operated the interfaces, initiated actions and given responses, and the type of functionality performed

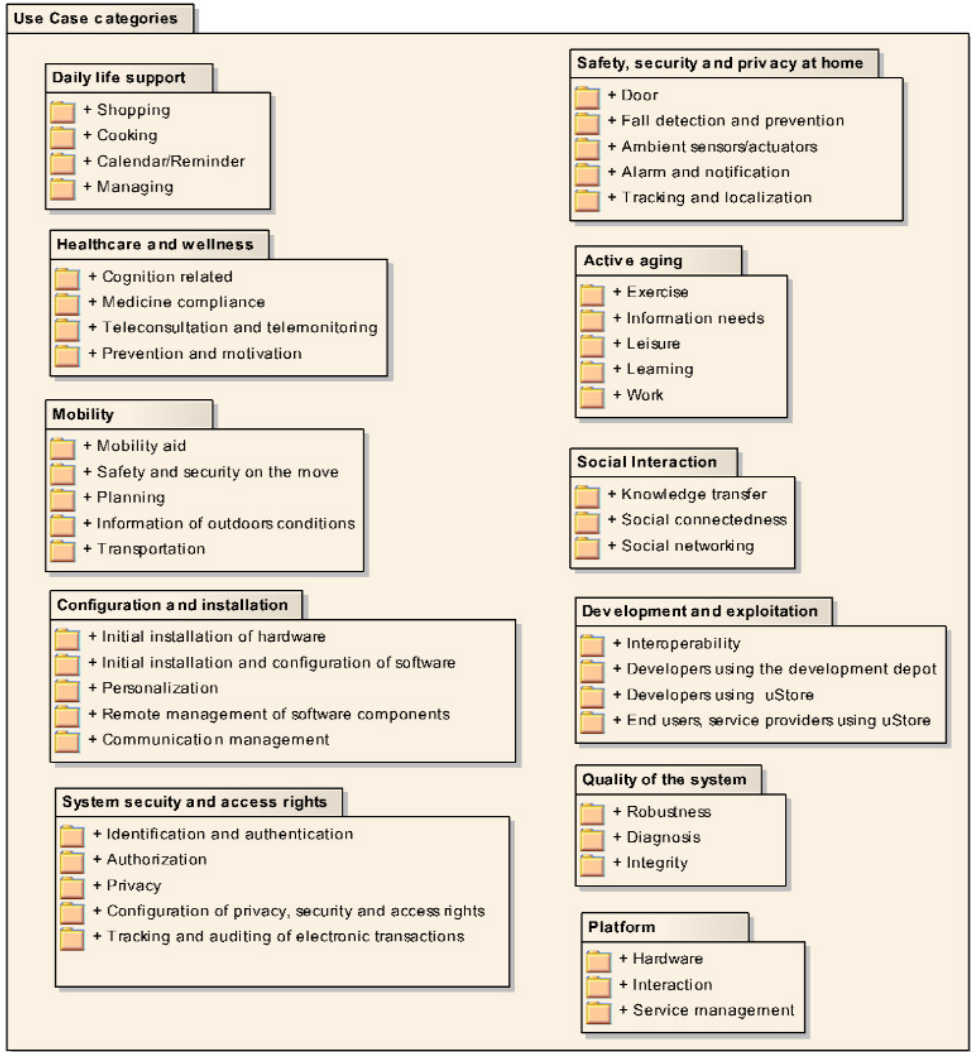
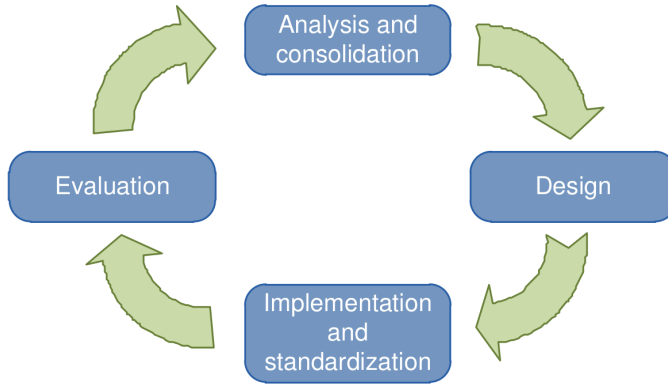


Figure 2.8. UniversAAL use case specifications

as a result of executed operations through UML sequence diagrams, UML use case diagrams and UML collaboration diagrams.

Figure 2.9 shows an overview of the iterative process applied in *universAAL* for requirement analysis. It consists of four iterations of refinement where the 1st iteration takes the requirements from the input projects as starting point and combines and consolidates them to a comprehensive list in a common format. Each further iteration takes the results from the previous iterations, analyzes possible drawbacks, and refines the list:

- *iteration 1*: for requirements, consolidation is understood as the collection of all requirements from the input projects, mapping them to each other, and finally merging



**Figure 2.9.** UniversAAL iterative process

them into one harmonized and prioritized list of requirements. However, *universAAL* has to also complement this process with a gap analysis, especially to cover the innovative aspects of the project work.

After gathering the statistics from the input projects, it was immediately clear that even having this focus on the above set of projects, the identified tasks of collecting, mapping, merging, harmonizing, and prioritizing the requirements cannot be done mechanically in a simple sequential way. Finally, the conclusion was to bring important aspects of the mapping and harmonization tasks into the collection phase in the sense that all the experts from the input projects that are contributing to the *universAAL* set of requirements have to agree on a target form of representing the requirements so that each of them does a mapping and harmonization of its own set of requirements in terms of the target form of representation.

The basic form was known to be a table, as this is the way *ARCADE* defines the requirement model. Hence, it was immediately clear that the agreed and common way of representing requirements in *universAAL* had to be related to the structure of this table. The first consequence was to agree on the set of columns in this table, which it is called the *requirement collection template*. Hence, by describing the input requirements based on this set of columns, requirements were automatically getting harmonized at a certain level.

The first consolidated list of requirements after the collection phase still suffered from: (1) missing data in the different columns because of the differences between the templates used in the input projects and the template used in *universAAL*, and (2) the different levels of abstraction and quality in the formulation of the requirements. These issues had made the task of merging similar requirements into one unique requirement almost impossible. One of the important initial works by the task force was to agree on the methodology for prioritizing the requirements. These groups of criteria were prioritized, according to which impact-related criteria and those related to the methodological evidence had higher scores.

- *iteration 2*: the objectives are: (i) narrowing the focus to reference architecture requirements. This was achieved by defining what reference architecture requirements shall look like and by selecting a set of reference architecture requirements from the set of consolidated requirements; (ii) categorizing and reformulating requirements to provide a list of requirements; (iii) analysing the requirements in relation to concerns in order to provide a consistent list of reference architecture requirements.
- *iteration 3*: the previous iteration provided the template but lacked in providing the detailed information for each requirement. Collecting all the properties of this template for each requirement was the main work done in this iteration.

Furthermore, a mapping between the requirements from the current list and the requirements from the list as defined in first iteration was created. This mapping information is part of the field “References” in the template. From this mapping, the information collected in the first iteration could be used; for example, the first iteration already specified a measure to check the solution against the requirement. This specification was reviewed and, if multiple requirements from the old list were found in the mapping, the specifications were combined.

A validation in form of a mapping between requirements’ categories and reference use cases (RUC) was performed. This is due to the fact that the requirements are needed to reflect the RUCs. During the mapping of RUCs to requirements it appeared that in most cases all requirements of a specific category were mapped to the same set of RUCs. Thus, it was decided to provide only a mapping between RUCs and categories instead of individual requirements.

- *iteration 4*: in *universAAL* experts’ groups deal with fields like security and privacy, the several functions of the middleware layer, remote interoperability, hardware abstraction, service orientation, user interface, configuration and context management. The purpose of the experts’ groups is to drive the development of the system in relation to the specific expert group topic.

Experts’ groups were created in discussions similar to the one that produced the new categories. These discussions identified the major technical topics of AAL that need to be addressed by an AAL reference platform. Therefore, the scope of the new requirement categories and the experts’ groups is strongly overlapping. Consequently, experts’ groups will deal with the refinement of the requirements and requirement categories that are related on a topical level.

## 2.2 Digital ecosystem

In the following, two of the most important implementations of *Digital Ecosystem* architectures are analyzed: the *DBE* [14] and *Meteor-S* [67] projects.

### 2.2.1 DBE project

The aim of *DBE* project is to provide an infrastructure to small and medium European sized enterprises (SMEs) able to make them more competitive through the adoption of

an information and communication program. This program allows SMEs to cooperate in the production of components according to local business needs.

*DBE* is inspired to living organisms mechanisms: evolution, adaptation, autonomy, viability, introspection, knowledge sharing, selection and it will lead to the emergence of novel architectures and technologies, business processes and knowledge. *DBE* wants to create an ecosystem where applications within it behave like intelligent, interactive and adaptive organisms.

The description of the model must be expressible in a natural language way but must also be computable and it must be searchable in order to support intelligent discovery. The semantics is coming from the *Model Driven Architecture (MDA)* [15] approach. A complete *MDA* specification consists of a definitive platform-independent (programming language and environment of execution) base model, plus one or more platform-specific models and sets of interface definitions (typically using *UML* formalism).

### **DBE architecture**

The *DBE* software project was architected [34] into three core systems: the *Execution Environment (ExE)* to host services, a *Service Factory* to design and develop services and an *Evolutionary environment (EvE)* to help to optimize the system.

#### *Execution Environment ExE*

The *ExE* system [24] is a *peer-2-peer* application container that isolates the programmer from the coding complexity. It hosts all the available services of the *DBE* providing a set of structural features like authentication, authorization, privacy, transactions, logging, and so forth.

It is composed by four main components:

- *the servant* that can be considered like a *DBE service container* that provides a gateway (*service adapter*) to existing services. The publication of the existence of service adapters consists on the creation of a *data item* which contains the service URL endpoint, the conforming models used by the service and additional user defined tags. The *data items* are then registered in a registry for service proxy called *FADA* that adopts a *Jini* [8] like protocol. If an external or internal module asks for a service, the servant searches for that acting as a client to the *FADA*;
- *the Service Factory* described later;
- *the locally accessible core components* that implement fundamental low level *DBE* functionality;
- *the networked addressable services* that are either infrastructural or SME specific.

#### *The Service Factory*

After the modelling phases, the *service adapters*, are created using the *Service Factory* module. The service factory is represented by a set of tools that allows organizations to

describe and publish their services using *DBE Studio* [21], *Eclipse* [53] based development environments and a decentralized repository to publish and retrieve service models and data. Each service in the *DBE* is specified using a set of formal languages that aims at defining the business models as well as the technical interfaces. The project has also developed a *Semantic of Business Vocabulary (SBVR)* [65], a *Business Modeller Editor (SBeaVer)* [23] for capturing business models and rules using natural languages. *SBVR* is a metamodel specification, adopted by OMG, for capturing business vocabularies and rules in a controlled natural language and representing them in formal logic structures. This approach has been demonstrated to be more effective with respect to the more common "boxes and lines" approach.

### *Evolutionary Environment EvE*

The *EvE* [22] is an optimization engine of the *DBE* and it is designed to monitor the consumption of services on the *DBE* for pointing out the one that is most likely to be consumed. It has also been designed to support the automatic composition of services, putting together atomic ones in order to deploy them as new to help satisfy potential users' needs.

### **DBE infrastructural services**

#### *Knowledge base*

The *knowledge base* service [66] provides the distributed storage facilities. Due to the distributed and dynamic nature of the SME-based network, the *knowledge base* replicates data following a primary / secondary asynchronous model in which one node is always the primary node for a particular piece of content. Should it fail, a secondary node becomes the primary. Content is replicated from primary to secondary nodes asynchronously. To improve the efficiency of queries, content is stored on nodes that already include semantically similar data. This is achieved by comparing the ontologies and semantics referred to inside the models. Then initial queries from a node are propagated to all nearby neighbors that store information regarding which nodes the results came from and a comprehensive set of routing information is built up to help direct future.

#### *Semantic registry*

The *semantic registry* [66] service is used to store published *Service Manifest*. A *Service Manifest* is an XML document that completely describes an individual *DBE* service. *Service Manifest* can be considered to be an advertisement for a service on the *DBE*.

#### *Distributed storage system*

The *distributed storage system* [51] delivers a generic distributed storage capability to the *DBE*. Essentially the *distributed storage system* allows arbitrary content to be persisted onto the *DBE* peer-2-peer network, and it generates an identifier by which the content can



later be retrieved from any node on the network. For redundancy, the content is replicated. To avoid the distributed system overfilling, all content must be assigned a time-to-live by the storing entity.

#### *DBE portal*

The *DBE portal* [32] is a core service which provides a user-friendly HTML interface *DBE*. This portal consists of a completely arbitrary website representing the SME's business. The *DBE portal* includes links pointing to the *DBE* services which that particular SME has deployed as well as the ability to search for arbitrary *DBE* services. *DBE portal* can also link to local *DBE* administration interfaces allowing basic servant configuration and functionality to be administered via web. The portal includes self-registration functionality which automatically publishes the existence of the portal service within the *DBE semantic registry*. Ultimately, this enables a peer-2-peer network of *DBE portal* to be formed.

#### *Recommender*

The *recommender* service [66] is an autonomous system that uses pre-configured user profile information to identify the best-matching *service manifest* published on the *semantic registry* that may be of interest. For example, if a user whose profile explains that he is interested in low-cost flights, he could be automatically alerted when a new low-cost flight booking service is published in the *semantic registry*.

#### *Habitat*

The *EvE* [22] is implemented in the *Habitat* service that is designed to support features such as autonomous service composition, the initial implementation uses neural networks to identify services that closely match those that have already been invoked.

#### *FADA*

*FADA* stands for *Federated Autonomous Directory Architecture* [81] and it is an open-source project originated in the European Commission. *FADA* was the first peer-2-peer infrastructure embedded into the *DBE*. *FADA* nodes find each other either by using broadcasting on a LAN or via manual configuration and they provide a location to store and retrieve proxies to services. If the *FADA* node does not have the requested proxy, it queries its neighbors and they, their neighbors, until the requested proxy is found. To avoid indefinite queries, maximum query times and number of hops can be specified. Whilst *DBE* uses *FADA* as a registry for service proxy, *FADA* also provides searching facilities whereby it is possible to assign tags, known as *entries*, to service proxies. *FADA* can then be queried to return not just proxies to specific services but also the proxies to all services that have been assigned with a certain *entries* too.

### 2.2.2 Meteor-S project

The growth of *web services* and *service oriented architecture (SOA)* offers attractive basis for realizing dynamic architectures. With the help of industry's wide acceptance of standards like *Business Process Execution Language for Web Services (BPEL4WS)*, *Web Service Description Language (WSDL)* and *Simple Object Access Protocol (SOAP)*, *Web Services* offer the potential of low cost and immediate integration with other applications and partners. The *Meteor-S* [67] project aims to extend these standards with *Semantic Web* technologies to achieve greater dynamism and scalability. The *Meteor-S* helps to provide plug-and-play support for dynamically selecting *web services* by enhancing discovery of relevant services using semantics. The project reduces manual intervention during Web process composition. The project has the ability of choosing the optimal set automatically [2].

#### The semantic

Four main semantic categories for the web process life cycle are individuated [75]:

- *data semantics*: formal definitions of input and output data messages of a web service. The semantic is needed for discovery and interoperability functions and it is performed annotating them using ontologies;
- *functional semantics*: are formally representing capabilities of a web service for discovering features. The composition of web service description is realized by annotating operations as well as providing preconditions and effects;
- *execution semantics*: are formally representing the execution or flow of a service in a process or operations in a service. They are used for analysis (verification), validation (simulation) and execution (exception handling) of the process.
- *QoS Semantics*: are formally describing operational metrics of a *web service* / process to select the most suitable service to carry out an activity in a process.

The *service description layer* of the *web service stack* provides the necessary information for invoking *web services*. *WSDL* is the *de-facto* standard for this layer. *WSDL* descriptions are syntactic and do not explicate the semantic of the service providers. For this reason, *Meteor-S* advocates semantic annotation of *WSDL* in two ways:

- *annotated WSDL 1.1*: it is a *WSDL* document with semantic features added to it via permissible extensibility elements present in the language. *Meteor-S* contains tools for manual annotation of *WSDL* or java source code as well as a schema matching based approach for semi-automatic annotation of *WSDL*;
- *WSDL-S*[3][70]: it is a semantically enriched *WSDL 2.0* document. It can be used the preferred semantic language (OWL, UML, WSMO) as well as a combination of them. enhancing web services description and discovery to facilitate composition.

### The components

The *Meteor-S* is composed of five elements:

- *abstract process*: it is like a *BPEL4WS* process with assigned semantic annotation. The component allows users to create the control flow of the process using *BPEL* constructors, to annotate each call to *web services* which require late binding and to specify process constraints / objectives from local and global optimization;
- *semantic web service developer / annotator*: the annotation process of *WSDL* described;
- *semantic publication and discovery engine*: it adds semantic extensions to *UDDI*. The ontology based semantic annotations are used to provide semantic matching based on subsumption and property matching. This tool allows users to publish semantically annotated Web services. Users can also employ a template based *GUI* or the discovery *API* to query the engine for matching services;
- *constraint analyzer*: it dynamically selects services from candidate services which are returned by the discovery engine. This selection is done on the basis of global *QoS* constraints and objectives for the process as well as domain constraints. The *QoS* optimization uses an keywordInteger Linear Programming solver and the *SWR* algorithm;
- *execution environment*: it consists of a binder and an execution engine. The binder performs actual late binding of services returned by the constraint analyzer and converts abstract *BPEL* to executable *BPEL*.

## 2.3 Interoperability using semantics

### 2.3.1 Introduction

To establish interoperability among different systems, there are at least three aspects that must be tackled:

- business processes;
- data models (semantic, syntax, including product classification and identification);
- communication protocols.

As regards interoperability, the communication protocols are not affected by sector specific issues and they are not so critical, in general they can come from other applicative domains and they can be directly applied, with no adaption. On the contrary, business processes and data models are very sector specific and if they try to directly transfer tools and solutions from other sectors, this is often unsuccessful.

### 2.3.2 eBiz

#### Introduction

The European project [69] aims to enable interoperability among small, medium and large enterprises among to the *Textile / Clothing and Footwear (TCF)* sectors, encouraging technology suppliers to provide better support and new advanced services for e-Business in order to create a reference architecture for the sector. The general key features of this architecture are (Figure 2.10):

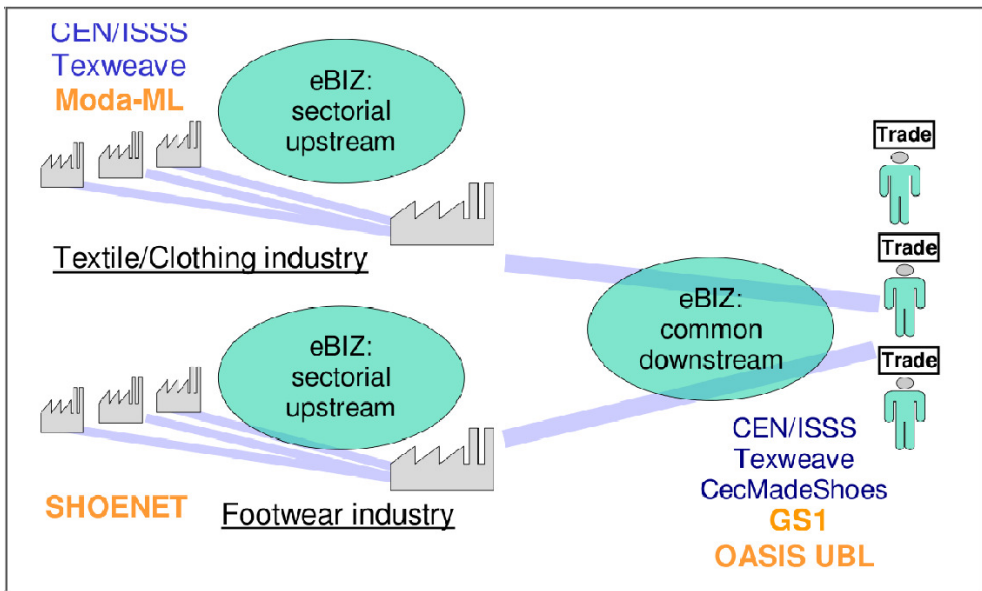


Figure 2.10. The eBiz architecture

- the exploitation of past experiences and advices of the different communities of users as *Moda-ML* [18] for the textile / clothing manufacturing, *Shoenet* [29] for the footwear industry, *OASIS* [26] and *GS1* [55] for industry and the generic technology *Electronic Data Interchange (EDI)* [31] for retailers;
- an inclusive open approach designed to support many models and communication solutions (*ASP*, *P2P*, *Hub*...);
- the use of public and usable specifications to reduce the gap between standard experts and company managers and technicians;
- the use of scalable architecture because it can be applicable to small and to medium enterprises as well as to large organizations;
- studied to be targeted to e-business real needs considering the different relation requirements that exists inside manufacturing networks, between producers and sellers and among retails themselves.

In this project was developed a shared semantics derived from standard specifications of *Moda-ML/TexWeave* [82], *ebXML* [25] and methodologies and tools were developed to facilitate their adoption. As an outcome of this work a *Knowledge Exchange Infrastructure (KEI)* was created. *KEI* is a conceptual framework supported by artifacts, software tools and a sectorial ontology [61] that, subsequently, has been published to support both semantic reconciliation applications as well as document (re-) engineering. *TFC* sectors are characterized by a large presence of SMEs and by an average level of adoption of e-Business and interoperability standards that appears to be quite low comparing to other similar manufacturing sectors. Innovative e-collaboration combined with other new manufacturing and supply chain paradigms can provide some of the answers to the European companies to strengthen or re-gain global competitiveness.

The specifications of e-Business for retail organizations are based on more generic *EDI* and in particular on *WWS Profile* and XML data model called *Universal Business Language (UBL)* [46]. A specific issue addressed in the architecture is the need for logistics and point of sale to have a common method to identify products and parties (companies and locations). The use of various and proprietary coding systems adopted by producers and retailers can present very hard problems of communication. To overcome this problem *GTIN* [39] and *GLN* [54] numbers, managed by *GS1*, provide identification for products and locations respectively and are internationally recognized and widely accepted by retail organizations in the world of consumer products. The need for a common product identification is not applicable to networks of manufacturers who deal with materials, components, processes and finished products that are easily identifiable as belonging to a specific producer.

### The architecture

The reference architecture includes two different areas:

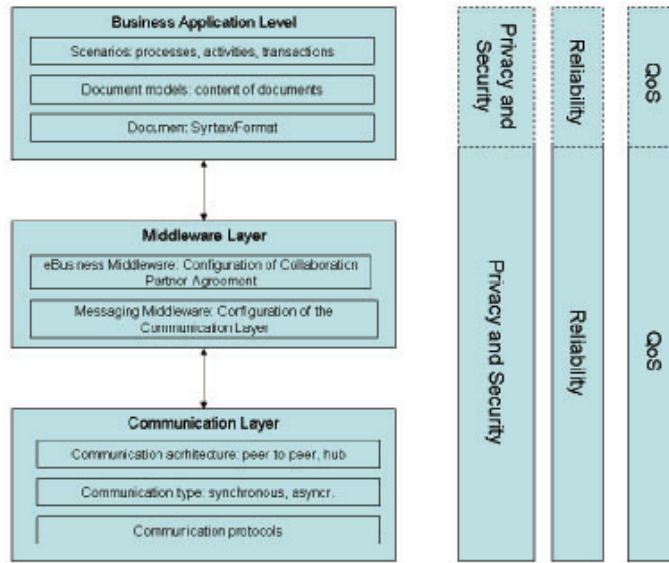
- *the high specialized networks of manufacturing enterprises (upstream area)*: the producers of final goods rely on complex networks of enterprises with highly specialized processes;
- *the retail channels for the Textile/Clothing and Footwear final goods (downstream area)*: the retail organizations need to achieve a common and efficient connection with the producers.

The project architecture is composed by three distinct layers (Figure 2.11):

- *business / application*: it is based on sectoral standards;
- *middleware*: it is based on *ebXML CPPA* [27] and *ebMS* [28]. It creates a connection between the upper and lower layer;
- *communication*: it specifies the type of connection and the underlying protocols.

In addition there are a Security / Privacy block and other vertical ones which can be applied on each of the hierarchical layers.

The architecture is based on four different types of specifications:



**Figure 2.11.** The eBiz architecture

- *business processes*: they are represented using *UML* notation and *ebBP* [25] templates;
- *data models*: document template specifications, based on a logical and syntactic level, implemented on XML but related with pre-existing EDI specifications;
- *collaboration and communication protocols*;
- *product classifications*.

### *Business application layer*

This layer describes all scenarios, processes and activities of the possible transactions among different actors of the business system.

As regards *TCF* sector, different standards related to data models exists and they can be classified in two categories:

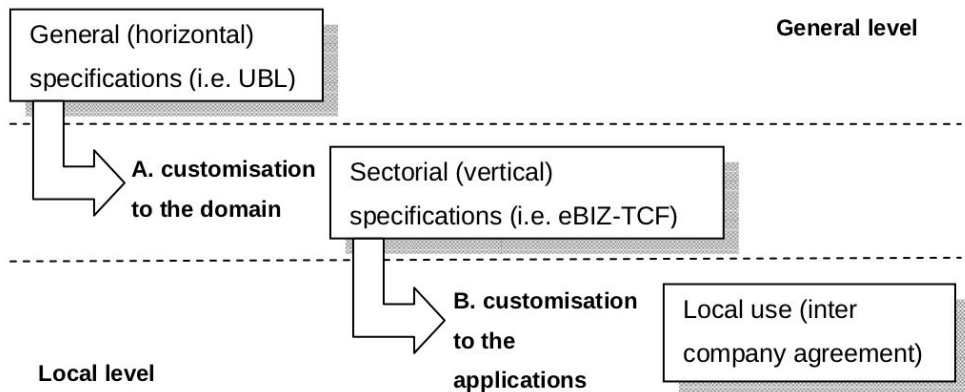
- *horizontal*: as *UBL*, *GS1*, *XML*. They are inter-sectoral standards and aim to cover a generic set of processes and data;
- *vertical*: as *TexWeave*, *ModaML*. They are closed in a specific domain.

This means that horizontal standards have specifics that are partially used in a realistic scenario and the vertical ones try to furnish a data model designed to manage exchanged information in a specific business domain. In real supply chain networks, the enterprises, need to define constraints that are stricter than the vertical standard specifications to reflect the requirements arising from very specific and dynamic businesses. In particular, two aspects are critical:

- the mapping between internal processes and data models into the standardized models;
- the reconciliation between different implementations of the same standardized specifications managed by different organizations.

Vertical standard appears to be more focused and effective to support real e-Business, but the problem is that certain industrial domains lack a sectoral standard to adopt for their e-Business transactions. Both using a horizontal and a vertical standard, at different levels, one of the problems is to define a trade-off between generality and complexity of specifications. To solve the problem, a solution consist in using *UML*, the most known modeling language. *UML* provides rules for the definition of profiles intended as extensions of the modeling language in order to make it fit to specific application domains. The possibility to define use profiles exists also for the *EDI* standard and they are called *EDI* subsets. These subsets are tailored for specific industrial sectors, but the specifications are released only in terms of hard paper manuals, without a clear and simple machine readable format that could ease their adoption. Another possible approach is based on the semantic reconciliation through a domain ontology without relying on a standardized specification but it is not taken into account because the largest part of the *IT* suppliers of the industry were not able to manage such kind of technologies while the more diffused *XML* based technologies are accepted and recognized.

According to this analysis, in business document modeling we envisage an approach not only to support the reduction form a general horizontal specification (*UBL 2.0*) to a sectoral scenario without losing the conformance with the standard, but also to simplify the process of implementation and to limit the complexity of *UBL* documents for the final users; this approach leads to the definition of a use profiles (Figure 2.12)



**Figure 2.12.** The eBiz use profile definition

#### *Middleware and Communication layers*

Middleware and Communication layers are composed by:

- *e-Business Middleware* to formalize the agreements among collaboration partners in such a way to allow automatic configuration of the underlying Messaging Middleware Layer. This layer can provide also for additional services such as data format and content transformations, business process management (process integrity control, exception handling, error handling);
- *Messaging Middleware* allows for automatic configuration of the communication layer and on top of it provides for additional services such as routing, message reliability, security-related services etc.;
- *Communication* to physically transport the messages is based on Internet protocols (*HTTP* for synchronous communication and *SMTP* for asynchronous communication).

In particular (Figure 2.13):

	ebXML	Web Services	SMTP/POP
E-Business Middleware	ebCPPA+ ebBP	BPEL WSDL UDDI	
Messaging Middleware	ebMS (SOAP with attachements)	SOAP (Web Services)	SOAP with attachements
Communication	SMTP	HTTP	SMTP

**Figure 2.13.** The eBiz recommended standards

- *SMTP / POP* based approach implementing full EDI over internet;
- *ebXML* based approach corresponding to XML (instead of EDI) e-Business;
- *Web services* based approach, corresponding to advanced distributed computing paradigms in the context of e-Business.

### 2.3.3 Kassetts

#### Introduction

The project pursues the ambitious objective to facilitate the information exchange between actors in a multicultural and multilingual environment. To obtain this result it is necessary to analyze and classify the specific business documents of the logistic domain



and extract from them a common reference data model and vocabulary for automatic document translation.

Kassetts objective is to facilitate the information exchange among actors of a multi-cultural and multi-language environment. To reach this purpose is needed to analyze and classify the specific business documents of the logistic domain and extract from them a common data model and a vocabulary for automatic translations.

### *Metodology*

The output is build following these steps:

- the identification of logistic documents of all the countries of interest;
- formalization following a simple data model schema of the choose documents in the previous phase, identifying each relevant term and each description in the local and the English language;
- extraction of Business-to-Business taxonomies from the *SEAMLESS* ontology considering ISO standards, the currency, the language and the geo-references;
- extraction and drafting of data models of profiles of all companies from the *SEAMLESS* ontology, in local and English language;
- extraction of data models for the drafting do Business-to-Business documents. The model is the core of the ontology and it is derived from the *Universal Business Language (UBL)* and *Texweave* [82]. The model was accepted by all partners of the project and it models documents regarding the request for estimate, the estimate itself, the order, the state of the order, the delivery notification and invoice;
- the definition of a first version of a logistic vocabulary that includes relevant terms for the exchange of information among system actors. The vocabulary is created using logistic and business documents;
- analysis of the documents to identify things in common and to classify them according to categories;
- specification of the first embryo of the data model *KASSETTS*.

Communication and collaboration across the *SEAMLESS* system is based on the existence and on the relationships between a global ontology and the more general regional / sectoral and local ones. The Global Ontology (GLOB) includes all the concepts that are used to exchange information within the network and it can be thought as the union of things in common between the ontologies adopted by different mediators. The concepts are developed and described in English choosen as a lingua-franca.

### **2.3.4 Moda-ML**

*Moda-ML* [18] is an initiative that aims to publicly offer tools and format specifications for the data exchange based on *XML* and Internet and to stimulate convergence and consensus of associations and firms. It is the most powerful interoperability language for the Textile-Clothing industry from yarn to apparel manufacturing. Since its first release in

2003 it has constantly progressed in quantity and quality to provide for the needs of an ever extending demand of Users. At present it covers:

- 10 Business processes;
- 22 types of Companies of the *T/C* sector,

with the support of:

- about 80 e-document models (*XML Schema*);
- about 700 vocables (*XML Types*).

The approach of *Moda-ML* is based on the exchange of standardized *XML* documents (messages) that are publicly available via Internet.

The transmission of the *XML* messages is based on the *ebXML* protocol that is publicly available. *Moda-ML* participated the *CEN/ISSS* initiative *TexSPIN* and *TexWeave* to create a sectorial European standard for the B2B exchanges leaded by the *European Association of the Textile and Apparel industry, Euratex*.

*Moda-ML* began thanks to the European support to the *Moda-ML* project (*IST Take-Up Action Line IV.2.5 "Computing, communications and networks take-up measures"*) through the *V Framework Research Programm (cluster of projects Eutist-AMI)*. After the conclusion of the initial contract, *Moda-ML* continues through the activities of the *Technical Group* and of the *Pilot Users* of the project in other initiatives with regional, national and European support. In 2008 the *DG Enterprise and Industry* project *eBIZ-TCF*, aiming at harmonizing e-Business in *Textile Clothing and Footwear* industry, adopted the *Moda-ML* specifications to support industry networking.

## 2.4 Semantics for Domotics

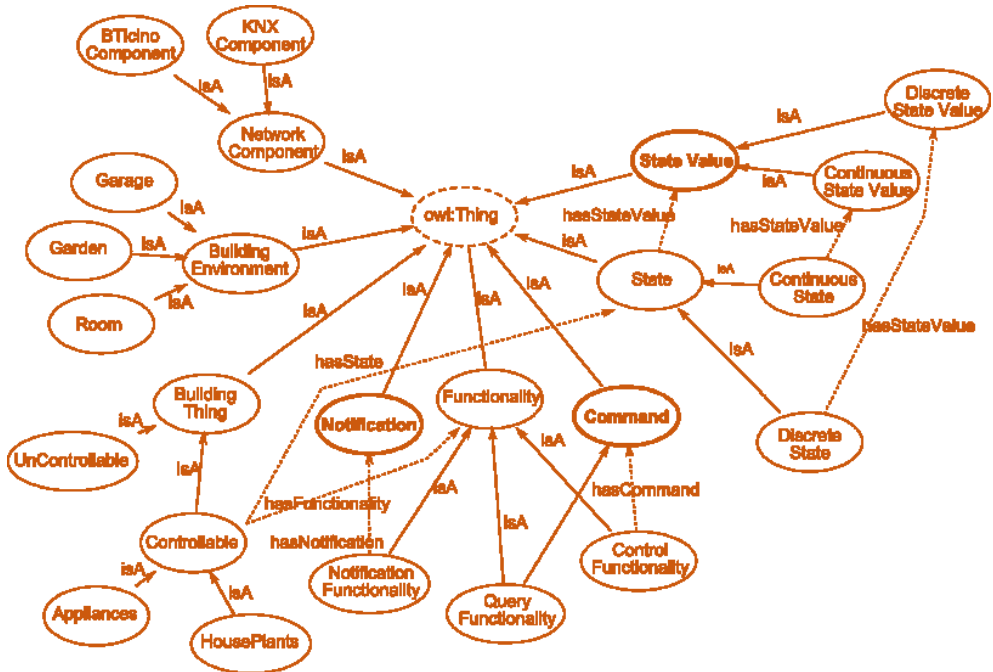
### 2.4.1 DogOnt

*DogOnt* is a novel modeling language for *IDEs (Intelligent Domotic Environment)*, based on *Semantic Web* technologies. By adopting well known representations such as ontologies and by providing suitable reasoning facilities, *DogOnt* is able to face interoperation issues allowing to describe:

- where a domotic device is located;
- the set of capabilities of a domotic device;
- the technology-specific features needed to interface the device;
- the possible configurations that the device can assume;
- how the home environment is composed;
- what kind of architectural elements and furniture are placed inside the home.

This information can then be leveraged by inference-based intelligent systems to provide advanced functionality required in *Intelligent Domotic Environments*. *DogOnt* is composed of two elements: the *DogOnt ontology* (Figure 2.14), expressed in *OWL*, which

allows to formalize all the aspects of a IDE, and the *DogOnt rules*, which ease the modeling process by automatically generating proper states and functionalities for domotic devices, and by automatically associating them to the corresponding device instances through semantic relationships. *DogOnt* is currently adopted to provide house modeling and reasoning capabilities to a domotic gateway called *Domotic OSGi Gateway (Dog)*, which is under development in the authors' research group and that will be distributed as open source toolkit for building *IDEs* running on low cost PCs. In this context, a third component of *DogOnt*, namely *DogOnt queries*, supports runtime control of the *IDE*.

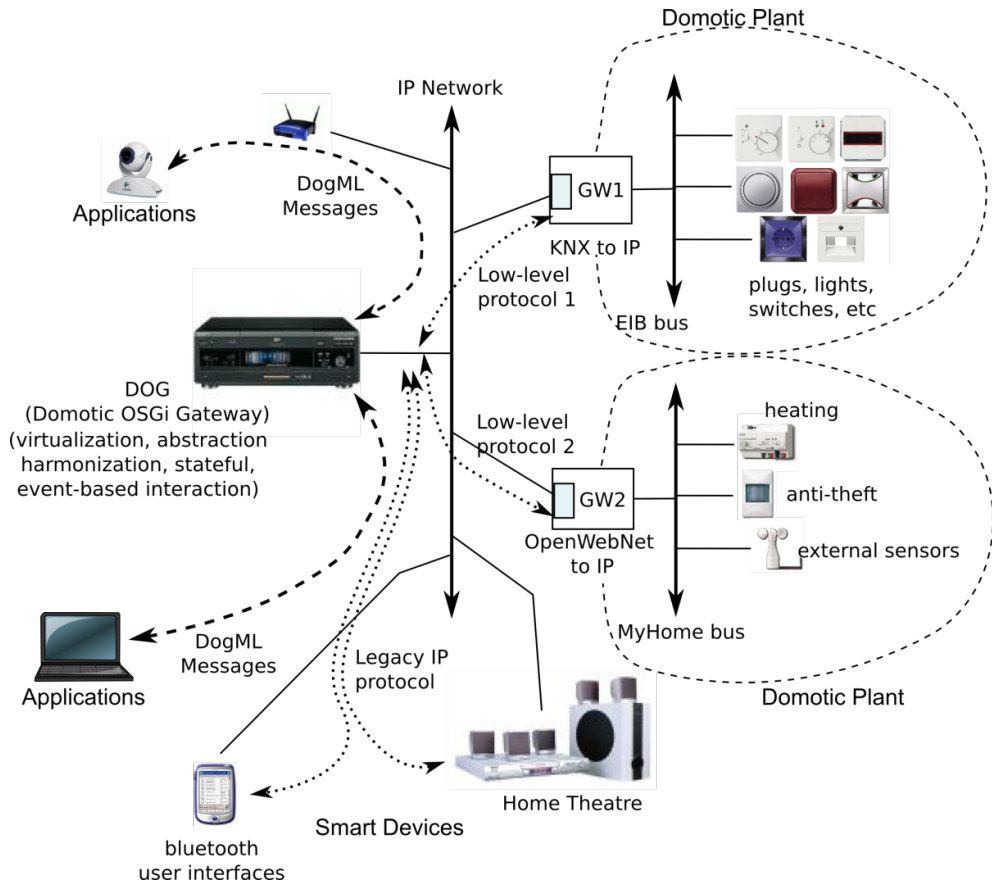


**Figure 2.14.** The DogOnt ontology

The *Domotic OSGi Gateway* is able to expose different domotic networks as a single, technology neutral automation system (Figure 2.15) using the *DogOnt ontology* to model devices and house environment. *Dog* provides the ability to control different devices installed in a home environment and to query different device properties ranging from location to current operating state.

### 2.4.2 SensorML

The primary focus of the *Sensor Model Language (SensorML)* is to provide a robust and semantically-tied means of defining processes and processing components associated with the measurement and post-measurement transformation of observations. This includes sensors and actuators as well as computational processes applied pre- and



**Figure 2.15.** The DogOnt architecture

post-measurement. The main objective is to enable interoperability, first at the syntactic level and later at the semantic level (by using ontologies and semantic mediation), so that sensors and processes can be better understood by machines, utilized automatically in complex work-flows, and easily shared between intelligent sensor web nodes. This standard is one of several implementation standards produced under OGC's *Sensor Web Enablement (SWE)* activity. This standard is a revision of content that was previously integrated in the *SensorML version 1.0* standard (OGC 07-000).

In its simplest application, *SensorML* can be used to provide a standard digital means of providing specification sheets for sensor components and systems.

By using *SensorML* it is possible to accomplish (Figure 2.16):

- *discovery of sensors, sensor systems and processes*: sensor systems or processes can make themselves known and discoverable. *SensorML* provides a rich collection of *metadata* that can be mined and used for discovery of sensor systems and observation processes. This *metadata* includes identifiers, classifiers, constraints

(time, legal, and security), capabilities, characteristics, contacts, and references, in addition to inputs, outputs, parameters, and system location;

- *lineage of observations*: *SensorML* can provide a complete and unambiguous description of the lineage of an observation;
- *on-demand processing of Observations*: process chains for geolocation or higher-level processing of observations can be described in *SensorML*, discovered and distributed over the web, and executed on-demand without a priori knowledge of the sensor or processor characteristics. This was the original driver for *SensorML*, as a means of countering the proliferation of disparate, stovepipe systems for processing sensor data within various sensor communities. *SensorML* also enables the distribution of processing to any point within the sensor chain, from sensor to data center to the individual user's PDA. *SensorML* enables this processing without the need for sensor-specific software;
- *support for tasking, observation, and alert services*: *SensorML* descriptions of sensor systems or simulations can be mined in support of establishing *OGC Sensor Observation Services (SOS)*, *Sensor Planning Services (SPS)*, and *Sensor Alert Services (SAS)*. *SensorML* defines and builds on common data definitions that are used throughout the *OGC Sensor Web Enablement (SWE)* framework;
- *Plug-N-Play, auto-configuring, and autonomous sensor networks*: *SensorML* enables the development of plug-n-play sensors, simulations, and processes, which can seamlessly be added to Decision Support systems. The self-describing characteristic of *SensorML*-enabled sensors and processes also supports the development of auto-configuring sensor networks, as well as the development of autonomous sensor networks in which sensors can publish alerts and tasks to which other sensors can subscribe and react.
- *archiving of Sensor Parameters*: *SensorML* provides a mechanism for archiving fundamental parameters and assumptions regarding sensors and processes, so that observations from these systems can still be reprocessed and improved long after the origin mission has ended. This is proving to be critical for long-range applications such as global change monitoring and modeling.

*SensorML* was approved by *OGC* as an international, open Technical Specification on June 23, 2007. UAH and other within a wide variety of sensor communities are in the process of building Process Models, Process Chains, documentation, and software in support of *SensorML*. Much of this will be Open Source, while some will be within Commercial Software. *SensorML* has been under serious consideration and testing by a wide range of communities, including those in the science, defense, intelligence, and public sectors.

*SensorML* is currently encoded in *XML Schema*. However, the models and encoding pattern for *SensorML* follow *Semantic Web* concepts of *Object-Association-Object*. Therefore, *SensorML* models could easily be encoded for the *Semantic Web*. In addition, *SensorML* makes extensive use of soft-typing and linking to online dictionaries for definition of parameters and terms.

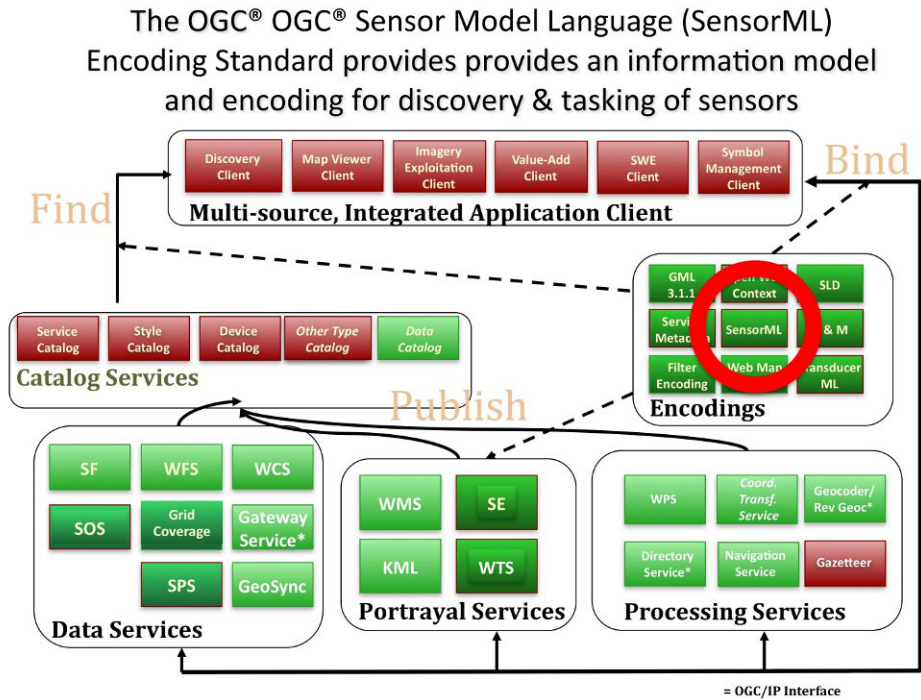


Figure 2.16. SensorML overview

## 2.5 Other works

Tin-Yu Wu et al. [89] propose an interesting work related the auto configuration of intelligent appliances in a networked domain. It implements three functions: (i) to assist the information appliance in acquiring a regular domain name without manual configuration; (ii) to provide session initialization protocol, uniform resource identifier, auto configuration and their registration; (iii) to initiate communication messages between devices, to manage the residential gateway and to configure the user management system interface. Unfortunately this solution requires to embed a program that runs during device system boot and this means to modify the original device design adding different routines to embed.

Another interesting work is presented by van Moergestel and Meyer [84]. They present a multi-agent based architecture for domotics to implement interoperability. The described architecture provides to each device an independent agent to which, an *IPv6* address should be assigned. Unfortunately, the solution requires the equipment of a small computer system for each device, where the corresponding agent is executed.

Ruta et al. [74] presented a distributed knowledge-base agent framework for home and building automation based on a semantic enhancement of *KNX* standard allowing the integration of knowledge representation technology and reasoning techniques. The pro-

posed approach supports advances, fine-grained resource/service discovery grounded on the formal annotation of user characteristics and device capabilities and exploiting logic-based negotiation. To support agent-based collaborative framework, the approach leverages a knowledge base evolution of *KNX* and particularly it implements a semantic micro-layer on the top of *KNX* protocol stack. Novel services and functions have been introduced while keeping a full backward-compatibility with the current protocol and devices. Thanks to such semantic enhancement to the standard, device features can be fully described by means of annotations expressed via logic languages. Unfortunately, this solution seems to work only for *KNX* domotic standard.





## Architecture

### 3.1 Introduction

The path undertaken in this research work is to supplement a monitoring platform with a software able to learn and anticipate the habits and behaviors of people in their own residence, and hence the ability to recognize possible worsening of health status such as abnormal behavior patterns signaling potential imminent health crises. The work presents a software architecture based on a semantic layer that enables the software platform to fully exploit knowledge-based representations of the environment. To this end, the software platform must be endowed not only with knowledge of the devices and their functions (e.g. knowing what a lamp, thermostat are and giving a meaning to switching them), but also with tools that enable it to learn concepts such as room, furniture, object position and the repercussions (effects) of the use of such objects. In this way, it is possible to assign to the software platform a specific goal to achieve without however having to specify every single action to perform. As an example, if the user wants more light in the environment, the software platform must autonomously decide to lift the blind or to switch the light on.

For these purposes, the semantic layer must provide solutions for:

- semantic interoperability among heterogeneous systems and devices, while masking their technological differences to users and external applications;
- enrich the user environments, domotic devices and their functions with semantic ability;
- defining the position of domotic devices and furnitures in order to communicate with smart objects using their position inside the environment (e.g. the lamp on the night-table, the TV in the kitchen);
- modelling the effects activating domotic functions in order to understand the changes that a device produces inside an environment.

To make the software platform able to recognize any abnormal user's behavior pattern and to anticipate his needs, the software platform must learn behaviors and habits of home inhabitants. This software must be adaptive, a context-aware application that works

in a fully interoperable environment. To this end, the software must perform real-time analyses to identify user's behaviors and verify and apply the rules learned.

### 3.2 Dictionary

This is a collection of the key words and their definitions that are used in this work:

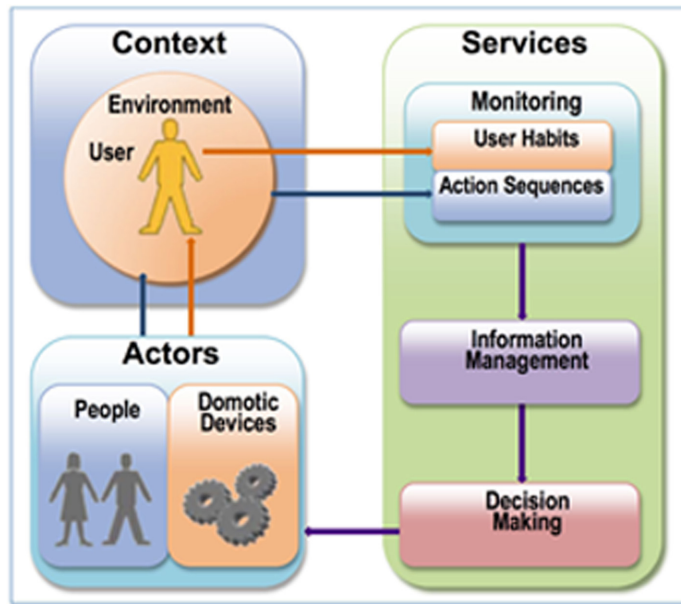
- *domotic function*: capability of the domotic device that permits it to change its state and to cause a modification in the environment and or a benefit for the user;
- *domotic system*: set of intelligent devices (sensors and actuators) and communication protocols belonging to a particular brand or standard and that are able to implement smart functionalities in the home environment. Each domotic system is often different and not interoperable with the others;
- *physical or domotic device*: the hardware of the device installed in the environment;
- *software platform*: product object of this study and development;
- *user action*: user interaction with a domotic device to activate one of its functions;
- *virtual device*: digital representation of a device that is physically installed in the environment.

### 3.3 Analysis requirements

The purpose of the requirements' analysis is to establish an understanding of the application domain and to capture, formalize, analyze and validate the user requirements on the software platform to be built. For this purpose the software platform is viewed as a black-box and only the objects and concepts visible on the boundary and outside the software platform are modeled.

The user interacts with the environment activating one or more functions available on the environment. A function can be the simple switch of a light, the interaction with the TV, stereo and so on. The interaction can take effect through a physical interaction, like acting on a switch button, or a remote interaction, like using a specific software in a tablet, smart-phone and so on. The interaction produces effects on the environment due to the changing of the state of some smart device (using the actuator of the lamp, this illuminates the environment; turning on the radio, the music is widespread etc.). Meanwhile, the activation of the functionality is captured by a sort of Environmental Manager that computes the information to eventually generate rules to let the software platform to autonomously interact with intelligent devices. As shown in figure 3.1, the lifecycle of an *Aml*-based system includes the acquisition of information about users and their environment by means of a *monitoring service* software module. The collected data is then analysed and processed by the *information manager* module. A *decision-making* software application then uses this processed information to identify the actions to be performed using machine learning techniques. Lastly, decisions are translated into commands and sent to recipient domotic devices, which together with any reaction on the

part of the occupants, modify the initial settings. The operation of the system can therefore be viewed as an infinite loop whose steps are performed in sequence each time that an update notification is received on the status of a device in the domotic network.



**Figure 3.1.** Hierarchy of actors

The software platform:

- is an application;
- is able to make interoperable domotic devices belonging to different systems and protocols;
- identifies and analyzes user actions;
- creates rules to anticipate user actions to satisfy user needs;
- sends commands to domotic devices;
- notifies behavior changes in user activities.

The user:

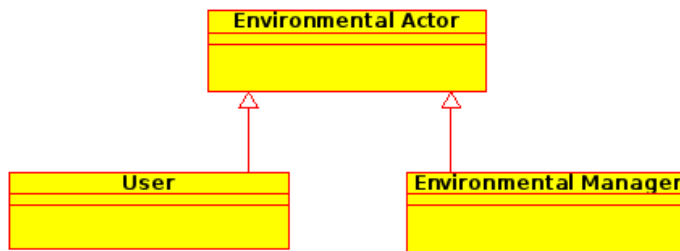
- interacts with the environment physically activating domotic devices and exploiting web, palm, smart-phone, tablet graphical interfaces to satisfy his needs.

### 3.4 Use Case

Use cases permit to describe the use of the software platform from the point of view of actors. An actor is a role played by a user or any other system that interacts with the subject. The primary purposes for these use cases are:

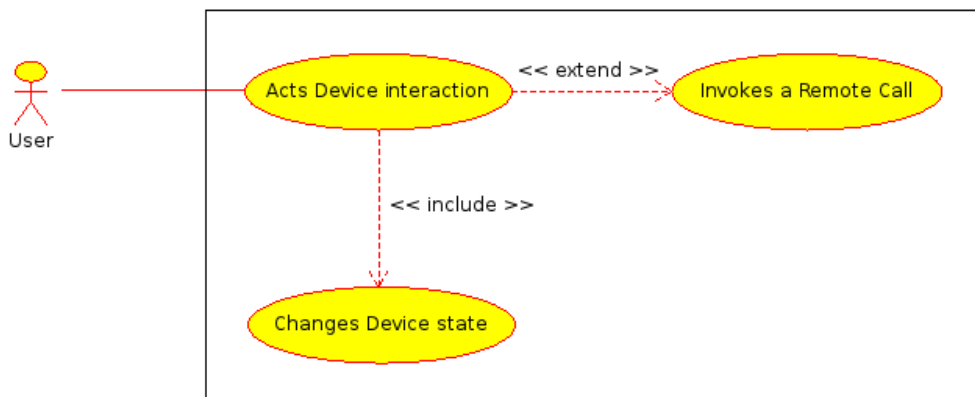
- to decide and describe the functional requirements of the software platform;
- to give a clear and consistent description of what the software platform should do;
- to provide a basis for performing software platform tests;
- to provide the ability to trace functional requirements into actual classes and operations in the software platform.

The actors of the software platform are classified as follow:



**Figure 3.2.** Hierarchy of actors

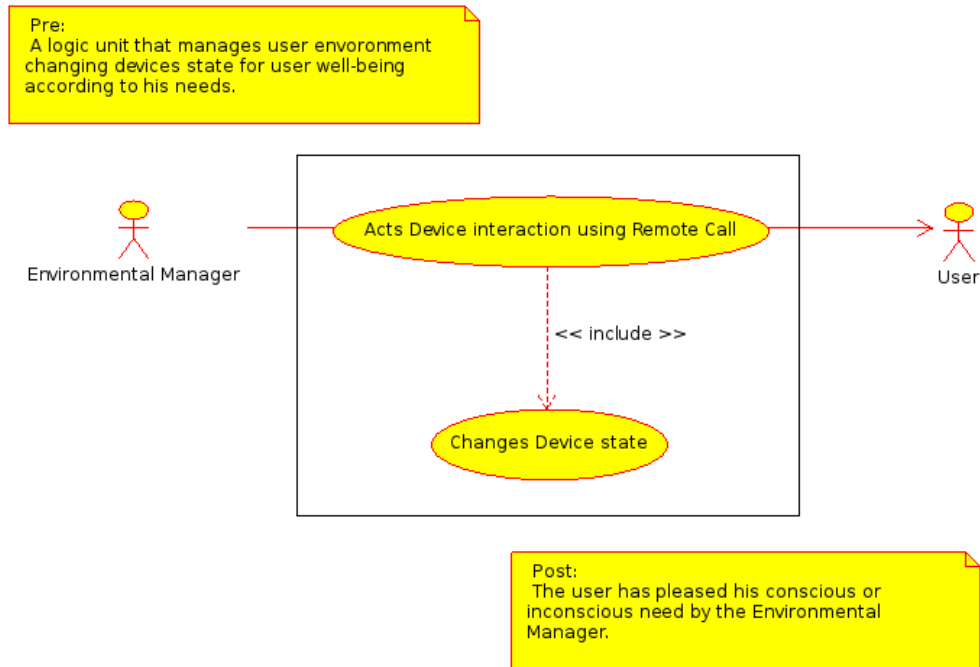
As shown in figure 3.2, an actor (*Environmental Actor*) is a entity that interact with the environment and it can be a human (*User*) or a software agent (*Environmental Manager*).



**Figure 3.3.** User use case diagram

As regards the human user (Figure 3.3), he can interact with the software platform activating a function on a device acting on it both physically (e.g. pushing a button) and using a remote interface (e.g. using the smart-phone to activate remotely a function). The

result of this interaction produces a change in the devices state and therefore, a change on the state of the environment.



**Figure 3.4.** Use case diagram

The software platform can decide by itself to act in place of the users (Figure 3.4) to satisfy a user's need. In this case, the *Environmental Manager* interacts properly with the correct device to activate the desired function able to change the state of the environment as needed. To do that, the *Environmental Manager* invokes a remote call to the software platform to control the device.

### 3.5 Network architecture

The architectural network model used to develop the software platform is inspired to the *Digital Ecosystem* paradigm. This paradigm permits to better model the issue. In fact, as shown in 1.2.4, the *Digital Ecosystem* is composed by two main elements: the *species* and the *environment*.

In this architecture, the *species* are represented by all the objects in the user environment like Domotic systems, furnitures, users, intelligent systems; the *environment* is represented by locations (e.g. rooms like kitchen, bedroom and so on.) that implement support services to permit species to live and to cooperate.

Each category of objects is controlled at least by one software agent (Figure 3.5).

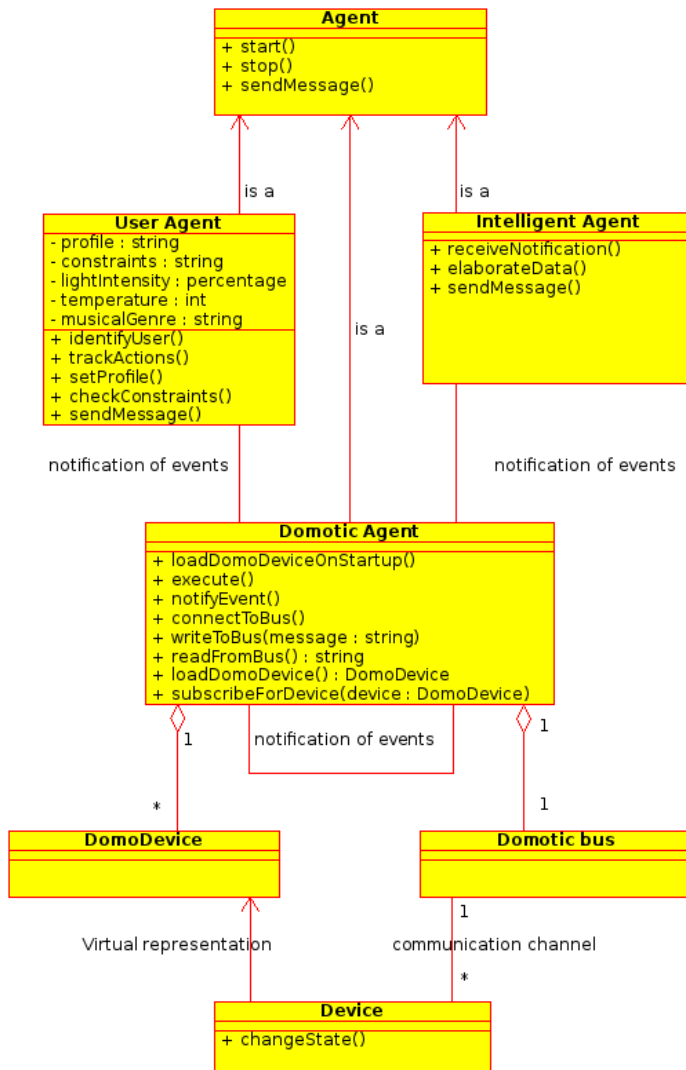


Figure 3.5. Agent class diagram

A software agent is a piece of software that functions as an agent for a user or another program, working autonomously and continuously in a particular environment. It is inhibited by other processes and agents, but it is also able to learn from its experience acquired by functioning in an environment over a long period of time.

Software agents offer various benefits to end users by automating repetitive tasks. The basic concepts related to software agents are:

- they are invoked for a task;
- they reside in "wait" status on hosts;
- they do not require user's interaction;

- they run status on hosts upon starting conditions;
- they invoke other tasks including communication.

### 3.5.1 Domotic agents

As shown in paragraph 1.2.5, in the market exist a lot of not interoperable domotic systems. The proposed solution foresees that each domotic system is managed by a specific and independent agent that connects the domotic bus with the software platform. This connection allows the software platform to share data, information and to manage devices inside the software platform to implement the needed interoperability and to operate with different domotic systems together.

Each *domotic agent* has to:

- open the connection with the domotic bus;
- read and write messages from / to the domotic bus;
- load at startup time, all devices belonging to the domotic system;
- implement interoperability features sending a command to other *domotic agents*;
- send notifications of events to other agents (e.g. to apply artificial intelligence algorithms, to compile statistics etc.).

In figure 3.5 is shown that a *domotic agent* is a subclass of the *Agent* class and it is composed by the domotic bus used as communication channel between the software platform and the real devices. The domotic bus permits to read information related to the events that occur in the domotic network and to write on it in order to execute commands.

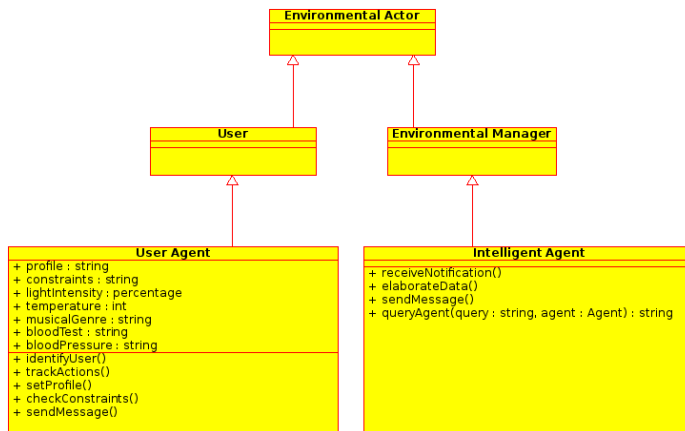


Figure 3.6. Intelligent Agent is an Environmental Manager

### 3.5.2 User agents

Each user is represented inside the software platform like an agent. Users are generally humans that live in a domestic environment where the software platform is running. The

agent is a sort of *alter ego* of the user and it represents him, his needs, preferences (e.g. like the preferred music genre, the ideal temperature in the house), and medical constraints.

The user interacts with the system platform using his virtual representation when it acts using the device physically (i.e. he pushes the switch button) or through a remote interface (like smart-phone, tablet etc.). A user identification is required to gather a set of actions to the correct agent.

Each *user agent* has to:

- identify the user;
- keep track of user's actions;
- set and modify the user profile according to user's actions;
- verify the defined preference and medical constraints;
- execute domotic functions sending commands to domotic agents.

As shown in Figure 3.6, due to their ability to interact with the environment, *User agents* are *Environmental Actors* and in particular, they are *Users*.

### 3.5.3 Intelligent agents

*Intelligent agents* are special agents that exploit intelligent behaviors to implement special functionalities and reach defined goals. They are able to collect information coming from the other agents and to process input data to take decisions and act consequently. Example of intelligent agents are those used for statistics, to implement energy saving measures, etc.

Since these kind of agents need to be customized, they are described here using a general description. Each *intelligent agent* has to:

- receive notifications from the other agents;
- query data of other agents to get necessary information;
- elaborate input data;
- send messages.

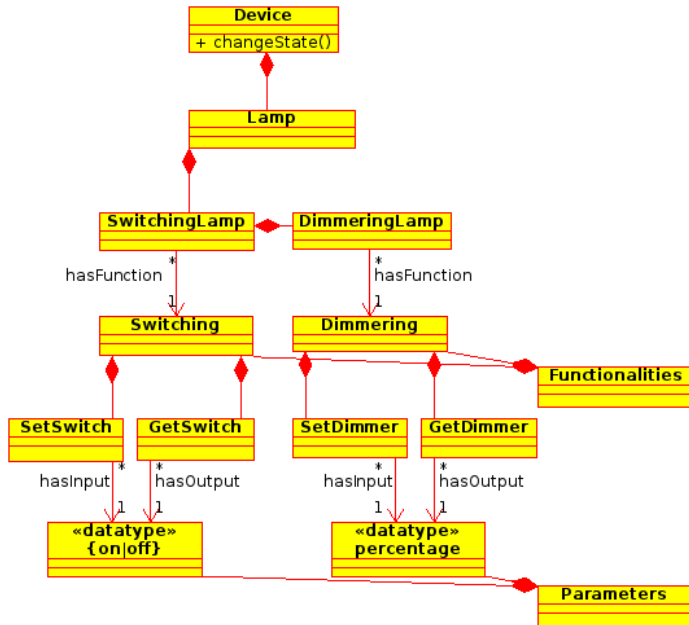
As shown in Figure 3.6, due to their ability to interact with the environment, *Intelligent agents* are *Environmental Actors* and in particular, they are *Environmental Managers*

## 3.6 Semantic approach

To abstract and represent devices inside the software platform and to implement interoperability, a semantic approach is applied. The semantic approach provides a semantic description of the operations of the home automation devices, the furniture, the environments and the lexicon that are supported by the platform.

All of the descriptions are formalized through a set of dedicated *Ontological Classes*, whose concepts are applicable and always true independently of the context in which





**Figure 3.7.** Ontological representation of the Lamp class, using UML

they are used. For example, figure 3.7 shows the semantic model of a *SwitchingLamp* class device and its functionalities: *SetSwitch* to set the light on or off using a Boolean value; *GetSwitch* to get the lamp's current status. *SwitchingLamp* class includes a *DimmeringLamp* class, which has the extra feature (over a "simple light") of being adjustable in its intensity. Such information is described in the *DomOnt* ontology (*Domotic Ontology*), which defines an integrated "taxonomy" of domotic devices and their functionalities. This ontology defines separate classes of devices, their functionalities, input/output parameters and the effects they have on the environment.

## 3.7 Implementation tools

### 3.7.1 Java

The *Software Platform* is implemented using the *Java* programming language. *Java* has gained enormous popularity since it first appeared. Its rapid ascension and wide acceptance can be traced to its design and programming features, particularly in its promise that you can write a program once, and run it anywhere. *Java* was chosen as the programming language for network computers (NC) and has been perceived as a universal front end for the enterprise database. As stated in *Java* language white paper by Sun Microsystems: "Java is a simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, multithreaded, and dynamic."

*Java* has significant advantages over other languages and environments that make it suitable for just about any programming task.

The advantages of *Java* are as following:

- *Java is easy to learn*: *Java* was designed to be easy to use and it is therefore easier to write, compile, debug, and learn than other programming languages;
- *Java is object-oriented*: this allows you to create modular programs and reusable codes.
- *Java is platform-independent*: one of the most significant advantages of *Java* is its ability to move easily from one computer system to another. The ability to run the same program on many different systems is crucial to World Wide Web software, and *Java* succeeds at this by being platform-independent at both the source and binary levels.
- *Java is distributed*: *Java* is designed to make distributed computing easy with the networking capability that is inherently integrated into it. Writing network programs in *Java* is like sending and receiving data to and from a file.
- *Java is secure*: *Java* considers security as part of its design. The *Java* language, compiler, interpreter, and runtime environment were each developed with security in mind.
- *Java is robust*: robust means reliable. *Java* puts a lot of emphasis on early checking for possible errors, as *Java* compilers are able to detect many problems that would first show up during execution time in other languages.
- *Java is multithreaded*: multithreaded is the capability for a program to perform several tasks simultaneously within a program. In *Java*, multithreaded programming has been smoothly integrated into it, while in other languages, operating system-specific procedures have to be called in order to enable multithreading.
- *Internationalization*: the *Java* language and the *Java* platform were designed from the start with the rest of the world in mind. *Java* is the only commonly used programming language that has internationalization features at its very core, rather than tacked on as an afterthought. While most programming languages use 8-bit characters that represent only the alphabets of English and Western European languages, *Java* uses 16-bit Unicode characters that represent the phonetic alphabets and ideographic character sets of the entire world. However *Java*'s internationalization features are not restricted to just low-level character representation. The features permeate the *Java* platform, making it easier to write internationalized programs with *Java* than it is with any other environment.
- *Performance*: as I described earlier, *Java* programs are compiled to a portable intermediate form known as byte codes, rather than to native machine-language instructions. The *Java Virtual Machine* runs a *Java* program by interpreting these portable byte-code instructions. This architecture means that *Java* programs are faster than programs or scripts written in purely interpreted languages, but they are typically slower than *C* and *C++* programs compiled to native machine language. Keep in mind, however, that although *Java* programs are compiled to byte code, not all of

the *Java* platform is implemented with interpreted byte codes. For efficiency, computationally intensive portions of the *Java* platform—such as the string-manipulation methods—are implemented using native machine code. Although early releases of *Java* suffered from performance problems, the speed of the *Java VM* has improved dramatically with each new release. The VM has been highly tuned and optimized in many significant ways. Furthermore, many implementations include a just-in-time compiler, which converts *Java byte codes* to native machine instructions on the fly. Using sophisticated *JIT compilers*, *Java* programs can execute at speeds comparable to the speeds of native *C* and *C++* applications.

- *Programmer Efficiency and Time-to-Market*: the final, and perhaps most important, reason to use *Java* is that programmers like it. *Java* is an elegant language combined with a powerful and well-designed set of APIs. Programmers enjoy programming in *Java* and are usually amazed at how quickly they can get results with it. Studies have consistently shown that switching to *Java* increases programmer efficiency. Because *Java* is a simple and elegant language with a well-designed, intuitive set of APIs, programmers write better codes with fewer bugs than for other platforms, again reducing development time.

Because of *Java*'s robustness, ease of use, cross-platform capabilities and security features, it has become a language of choice for providing worldwide Internet solutions.

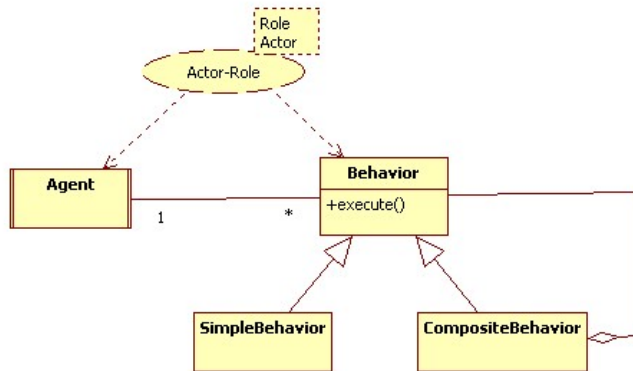
### 3.7.2 JADE

The structure of the architecture is based on the *JADE Platform*. *JADE (Java Agent DEvelopment Framework)* [10] is a software Framework fully implemented in the *Java* language. It simplifies the implementation of multi-agent systems through a middleware that complies with the *FIPA* specifications and through a set of graphical tools that support the debugging and deployment phases. A *JADE*-based system can be distributed across machines (which don't even need to share the same OS) and the configuration can be controlled via a remote GUI. The configuration can be even changed at run-time by moving agents from one machine to another, as and when required. *JADE* is completely implemented in *Java* language and the minimal system requirement is the version 5 of *Java* (the run time environment or the JDK).

Besides the agent abstraction, *JADE* provides a simple yet powerful task execution and composition model, peer to peer agent communication based on the asynchronous message passing paradigm, a yellow pages service supporting publish / subscribe discovery mechanism and many other advanced features that facilitate the development of a distributed system.

The communication architecture offers flexible and efficient messaging, where *JADE* creates and manages a queue of incoming *ACL* messages, private to each agent. Agents can access their queue via a combination of several modes: blocking, polling, timeout and pattern matching based. The full *FIPA* communication model has been implemented

and its components have been clearly distincted and fully integrated: interaction protocols, envelope, ACL, content languages, encoding schemes, ontologies and, finally, transport protocols. The transport mechanism, in particular, is like a chameleon because it adapts to each situation, by transparently choosing the best available protocol. Most of the interaction protocols defined by *FIPA* are already available and can be instantiated after defining the application-dependent behaviour of each state of the protocol. SL and agent management ontology have been implemented already, as well as the support for user-defined content languages and ontologies that can be implemented, registered with agents, and automatically used by the framework.



**Figure 3.8.** JADE behaviour model, using UML

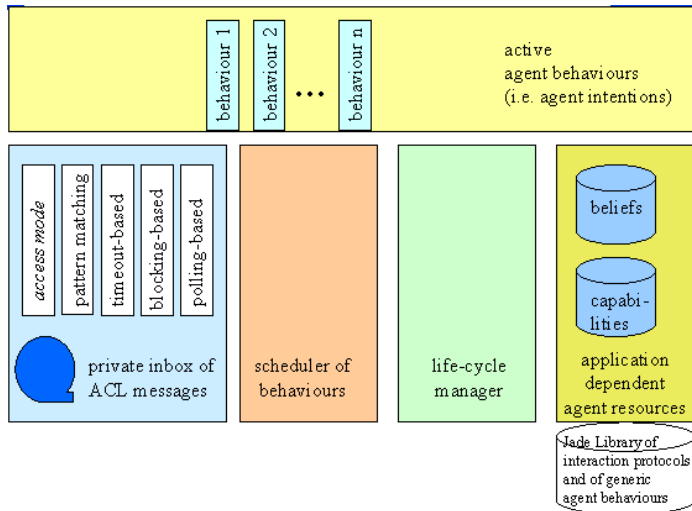
In Figure 3.9 is shown the architecture of a generic *JADE* agent. The actual job of an agent is typically carried out within "behaviours". A behaviour represents a task that an agent can carry out (Figure 3.8).

Thanks to the contribution of the *LEAP* project, ad hoc versions of *JADE* exist designed to deploy *JADE* agents transparently on different Java-oriented environments such as *Android* devices and *J2ME-CLDC MIDP 1.0* devices. Furthermore suitable configurations can be specified to run *JADE* agents in networks characterized by partial connectivity including NAT and firewalls as well as intermittent coverage and IP-address changes.

*JADE* is a free software and is distributed by "Telecom Italia", the copyright holder, in open source under the terms and conditions of the *LGPL* (Lesser General Public License Version 2) license.

### 3.7.3 OWLAPI

Ontologies for the *software platform* are developed using the *OWL* language (Paragraph 1.2.7) and the *Protégé* editor (Paragraph 1.2.7). To manage ontologies, the *OWLAPI*



**Figure 3.9.** Generic JADE agent architecture

library is used. The *OWLAPI* is an open-source *Java* library for the *Web Ontology Language* (OWL) and *RDF(S)*. The API provides classes and methods to load and save OWL files, to query and manipulate OWL data models, and to perform reasoning based on Description Logic engines. Furthermore, the API is optimized for the implementation of graphical user interfaces.

The API is designed to be used in two contexts:

- for the development of components that are executed inside of the *Protégé-OWL* editor's user interface;
- for the development of stand-alone applications (e.g., Swing applications, Servlets, or Eclipse plug-ins).

The *OWLAPI* [45] includes the following components:

- an API for OWL 2 and an efficient in-memory reference implementation;
- RDF/XML parser and writer;
- OWL/XML parser and writer;
- OWL Functional Syntax parser and writer;
- Turtle parser and writer;
- KRSS parser;
- OBO Flat file format parser;
- reasoner interfaces for working with reasoners such as FaCT++, Hermit, Pellet and Racer.



## Domotic agent

### 4.1 Introduction

As explained in the 3.5.1 paragraph, there is a *Domotic Agent* for each domotic system introduced in the software platform. The developed prototype includes two domotic agents for two supported domotic systems: *KNX* and *Universal Plug and Play (UPnP)*. These two domotic systems are particularly interesting when using them together because they are based on very different technologies. The first main and fundamental difference consists in the different approach method used to present devices inside their network. *UPnP* system is based on a *Plug and Play* approach. This means that when a device is inserted inside the *UPnP Network*, it presents itself exposing its functions for auto-configuration. A *UPnP Device* can join its network at any time without the need to restart the system. *UPnP* is used mainly for multimedia purposes and it uses the wireless network as a communication channel. A device belonging to *KNX*, in general, is not able to auto-configure and to present itself to the network. A setup procedure is needed. This procedure is performed using an external software application able to set device parameters to exploit its functions. In order to install physically a device, it could be necessary to shut down the network. *KNX* is used mainly for home automation purposes and it often uses a dedicated bus as a communication channel.

#### 4.1.1 UPnP system

##### Introduction

A growing number of embedded devices are connected to TCP/IP networks. And many of these devices are no longer passive network nodes, waiting for someone else to control them. They are full-blown network citizens, actively communicating with their peers and often relying on the network services provided by other devices to do their job. In order to work, all these devices must be configured properly. Configuring the network parameters (e.g., IP address, netmask, etc.) of a device is a tedious task, as many devices do not have an appropriate user interface to do this comfortably. This is especially an issue

with consumer devices, where the user might not even have the necessary technical knowledge to configure such a device. As the number of devices in a network grows, configuring each device separately is no longer practical. From this issue comes the need for the automatic configuration of network devices and the automatic discovery and invocation of network services. In recent years, the industry has come up with a variety of different technologies and specifications to address this problem. One of these technologies is *UPnP* - *Universal Plug and Play*.

### UPnP technology

*UPnP* technology offers pervasive peer-to-peer network connectivity of PCs, peripherals, consumer electronics and home automation devices. The *UPnP Architecture* is a distributed, open networking architecture based on *TCP-IP* and Web technologies like *HTTP*, *XML* and *SOAP* to enable seamless proximity networking in addition to control and data transfer among networked devices.

### The UPnP Device Architecture

The *UPnP Forum*, founded in 1999, is the industry initiative responsible for defining the specifications and standards for *UPnP*. In addition to the *UPnP Device Architecture*, which forms the basis of *UPnP*, the *UPnP Forum* defines device control protocols for specific devices' categories such as audio/video, home automation, networking, printing, etc.

The *UPnP Device Architecture* is the core of the *UPnP* and it defines the basic network protocol for communication among devices. The *UPnP Device Architecture* specifies six levels of protocols and technologies that must be implemented in a *UPnP* device. These are:

- addressing;
- discovery;
- description;
- control;
- eventing;
- presentation.

#### Addressing

Every device must be assigned a unique network address. In case of *TCP-IP* networks this can be done with the help of a *Dynamic Host Configuration Protocol (DHCP)* server. Should, however, no *DHCP Server* be available, another way of assigning an *IP Address* must be found. Apart from manual configuration, which is often undesirable, a method called *Automatic Private IP Addressing (APIPA, or Auto IP)* is used. In this case, a device's *TCP-IP Stack* randomly chooses an *IP Address* in the private range



from 169.254.0.0 to 169.254.255.255. To prevent two or more devices from accidentally selecting the same address, each device must probe, using the *Address Resolution Protocol (ARP)*, whether the chosen address is available.

### *Discovery*

A user or device must be able to find a service provided by one or more devices in the network. The important part here is that the user (or device) usually does not care which device implements the service, as long as the service with specific properties is available and accessible. A typical example for service discovery is: I need to print a document in color. Give me an *IP Address* and *Port Number* where I can send the print job to, using the *Internet Printing Protocol (IPP)*, so that the document will be printed in color.

What all technologies for service discovery have in common is, that they make use of *IP Multicasting*. *IP Multicasting* uses addresses in a special address range (224.0.0.0 to 239.255.255.255). A packet (typically, a *UDP Packet*) sent to a *Multicast Address* is received by all hosts listening to that specific address. *UPnP* uses *Simple Service Discovery Protocol* for that purpose.

Service discovery is implemented in the following way:

- an application or device that needs a certain service sends a request describing the required properties of the service to a specific *Multicast Address* (and *Port Number*);
- other applications or devices on the same network receive the request, and if they provide the requested service themselves (or know another device that implements the service), respond with a message describing where the service can be found;
- the application or device searching for the service collects all responses, and from the responses chooses the one service provider it is going to use.

In addition, devices that join or leave a network can send announcements to other devices describing the availability of the services they provide.

### *Description*

Once a certain service has been discovered, it may be necessary to obtain more information about the service. For example, if a service consists of multiple operations, it is necessary to find out exactly what operations are supported, and what arguments must be passed to them. This is the purpose of service description.

In case only well-defined service protocols are used (e.g., *HTTP*, *Internet Printing*, or media streaming), service description is not necessary, because the only information needed to access the service is the network address (*IP Address* and *Port Number*, or *URI*), and this information can be obtained by service discovery. In other cases, such as *UPnP*, the information obtained via service discovery may be insufficient to successfully access the service. In such a case, service discovery only returns the address of a network resource that provides detailed information about the capabilities of the service, and how to access them.

### *Control*

After an application has obtained enough information about the services it wants to access — either via service discovery alone, or together with service description, the application will access or invoke them in order to control the device. This is usually beyond the scope of most service discovery technologies, and the domain of specialized technologies and protocols. Examples for such technologies are *HyperText Transfer Protocol (HTTP)*, *SOAP*, *Remote Method Invocation (Java RMI)*, *Common Object Request Broker Architecture (CORBA)*, or media streaming protocols such as *Real Time Streaming Protocol (RTSP)*. *UPnP* uses *SOAP* for service invocation.

### *Eventing*

Eventing allows a device to notify interested parties about changes to its internal state, without requiring interested devices to actively poll its state. *UPnP* uses *GENA*, a protocol based on *HTTP* and *XML*, for eventing.

### *Presentation*

Presentation allows a device to display a user interface that can be used for monitoring or controlling the device by a human. *UPnP* capable devices use web technologies - *HTTP* and *HTML* - to implement the user interface. Thus, a web browser is necessary to display such a user interface. The web browser usually does not run on the device itself, but on a PC, smartphone or network-enabled TV.

## **4.1.2 KNX system**

### **Introduction**

The main concept of the specification is that of interworking which is a made up word to describe interactions around a standard message protocol. Unlike proprietary end-to-end solutions, where one vendor offers an integration solution and interacts with others on a one-off basis, here vendors implement the standard *KNX messages*. This opens up the end-to-end stack at each step of the interaction. For example panels from many vendors can interact with lighting from other, completely unrelated vendors. The advantage of this is an economical one in the market place. Where the mainframe was end-to-end solution with high prices (and margins), the PC offered modularity around an Operating System where unrelated vendors could participate in a solution by implementing the drivers. As a result there is a wealth of offering at every step of the solution. *KNX* creates a market-place. *KNX*:

- works over various twisted pair, power line (230V or low voltage bus), *RF* and *IP*. In the real world the only media supported is *TP* with low voltage;
- ISO/IEC 14543-3, ANSI + certain Europe specific standards and a Chinese standards;

- operation over *IP* or mobile available.

Covers:

- lighting;
- blinds and shutters;
- heating, ventilation and air control (HVAC);
- audio/video control (AV);
- operation and visualization;
- security;
- remote access.

### **KNX Group address**

The communication between devices is made with telegrams sent to *Group addresses*. A *Group address* is a logical name for a given *topic* linking the output of a sensor with the input of an actuator. The *topic* can be encoded in a *bit*, a *nibble*, a *byte*, *2 bytes*. These are considered signatures. Compatible devices listen in a group. One or more sensors emit the bits many actuators listen on a given group. This is the primary way to connect things in *KNX*.

For example, connecting the event "click on" is sent to a group we can call "lights on". The actuators are dragged-and-dropped in this group and when a message is sent with the group address on, the devices know to answer.

One can view a device, sensor or actuator, as a collection of objects. Whether these objects are exported, in a *Remote Procedure Call (RPC)* way is a configuration option. Once they are exported the output objects can be linked to the input objects by way of a communication *group address*.

A group address is represented as 2/3/4 (with slashes).

Is a group address the equivalent of Scene? yes and no. A group communication address is based on a type that is either *bit*, *nibble*, *byte* and one value, while a scene may involve several types (on bits and absolute dimming for example) and different values (one dimming at 25 and others at 75). So while programming groups that all respond to exactly the same input is a trivial task in *KNX*, a scene is a difficult, product dependent issue.

### **KNX Physical address**

A device always has a physical address. A *Physical address* is of the form *n/n/n* where the numbers represent a real topology. For example 1.2.4 (with dots). A *physical address* is allocated to each device during commissioning. A button needs to be pressed on the physical device the first time around for the commissioning to happen. Even on subsequent prints of the addresses the device requires to physically press the button. It makes sense that devices can be identified during installation. You can also ask for a given address to blink so you can physically identify a unit in the field.

The *Physical addresses* have an impact on topology. Because the devices are memory constrained, the numbers cannot be more than 64. This, it seems, means that the line segments need to have 64 devices and which has an impact on the electricians' work.

### KNX Specifications

KNX specifications include no less than 10 volumes.

Volume 3 - System Specification is the core of the spec, defining the reference for both hardware and software implementations.

The other volumes cover general architecture, information on specification tests and certification, conformance tests, device profile definitions, etc.

#### *Hardware and Networking Notes*

Depending on which media is used:

- *KNX Over IP*: *KNX specification* defines *KNX telegrams* which can be sent over *IP* using *UDP* (unicast and multicast). No additional equipment besides regular Ethernet hardware is required. 10Mbit Ethernet is sufficient. The communication stack is a standard IP stack (what we get from Linux with Java socket abstraction) + what is called the *KNX Common Kernel* on top of *IP* which basically handles the *KNX Telegram*  $\leftrightarrow$  *UDP translation*.
- *KNX Over RF*: No custom component required. *RF protocol* is defined in the *KNX specification*. *KNX/RF* doesn't seem to be used much. Since most devices are powered by the bus, *RF* does require independent powering. Bitrate 16kbps.
- *KNX Over Powerline (230V)*: Needs a bus coupling unit with the appropriate circuitry + comms stack on top of *ASIC* for *PL110*. Bitrate 1200bps.
- *KNX Over Twisted Pair*: Chipsets and comms stack available from *KNX system* vendors. Bitrate 9600bps.

### KNX Tools

*KNX Association* provides a collection of software tools for software developers, integrators and installers to ease the adoption of the platform.

- *ETS - Engineering Tool Software*: tools for configuring *KNX* certified products. *ETS* is independent of manufacturer, installer/integrator can use it to orchestrate all compliant devices over all different media. The *ETS* is licensed per PC. It uses *Falcon* as protocol stack. *ETS* is Windows only.
- *PC (Windows DCOM and .NET)*: software component implementing the *KNX* communication stack over *IP*, *RS232* and *USB*. It targets for software developers.

### Interesting implementations

Some open source or otherwise relevant *KNX* projects:

- *Calimero/KNX@Home*: Open source implementation (<http://calimero.sourceforge.net/> and <http://knxathome.fh-deggendorf.de/knxathome/wiki/index.php/KNXatHome>). A point of interest for OR is that *Calimero* abstracts away most of the network implementations for an application like the *ORC*. Access to the network becomes a simple *API programming* matter. It can be seen as a gateway to the *KNX bus*;
- *KNX USB Linux Driver*: User space module for Linux that bridges *USB* and *KNX* bus. *LGPL* license With a Simple *USB coupler* this is in essence a serial gateway to the *KNX bus*. The *API* seems less well documented than *Calimero*.

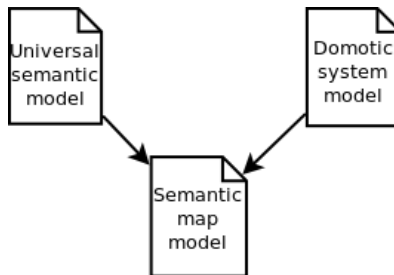


Figure 4.1. The ontologies used to represent devices

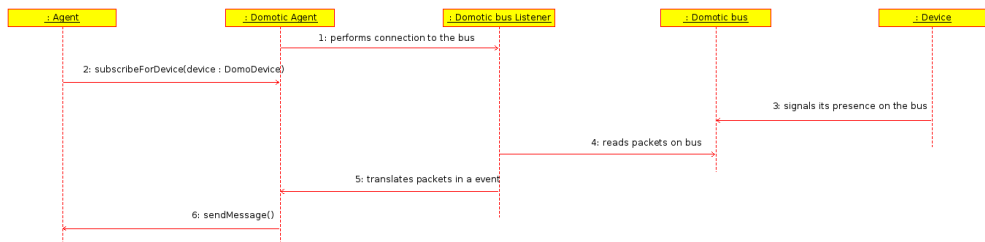
## 4.2 Interactions with domotic bus

For its functioning, the *Domotic Agent* has to establish a connection with the devices to be managed. In general, it is not possible and it is not expedient to interface directly devices but it is appropriate to do that interfacing their *domotic bus*.

A *domotic bus* realizes interconnections between the various home automation devices belonging to the same domotic system, introducing new features compared with a traditional wiring composed by individual components or small groups of them. In fact, while in the wiring of a simple electrical system of a house there is no distinction between communication lines and power lines, in a home automation system bus these two paths are decoupled. In the traditional electrical plant, each device is connected by a point-to-point wire and receives the information and the power on the same cable. The switch is used to determine the on / off state of the light bulb. If the two paths of the electrical power and information were decoupled, the pressure of the switch would mean only the signal of change of state of the bulb and in particular, the signal that would travel on the bus until the intelligent device which in turn would manage the transfer of necessary power to the bulb. The decoupling of the two lines has the enormous advantage of being able to add

new control points or new power devices without having to overturn the wiring, but simply by reprogramming the central control device. In addition, the digitization of information allows operations that are typical in a home automation system like the remote control.

Once the connection with the *Domotic bus* is established, the *Domotic Agent* is able to control domotic devices. To change the state of a domotic device, there is the need to have the ability to write packets on the *Domotic bus*. Writing packets in the *Domotic bus* is usually simple. Once the packets to send are built, it is sufficient to put them into the bus media. Instead, if there is the need to know the state of a domotic device, or to check when it changes state, it is necessary to have the ability to read from the *Domotic bus*. To read packets from the *Domotic bus* (Figure 4.2) can be a little more difficult than the writing process. Infact, there is often the need to provide a listener who waits for packets to pass through the *Domotic bus* to analyze them to find information of interest. The captured information can be used and sent to the other Agents.



**Figure 4.2.** The Domotic Agent reads an event from the bus

In the ideal case, at startup time, the *Domotic Agent* creates all the available *Domo Devices* depending on the possibility to discover the domotic network in that moment. The typical situation is when the Domotic network does not change after the startup time. Instead, if the domotic network can grow at run time, *DomoDevices* are created when they become available.

When the Listener captures an event that is related to the change of state of a domotic device, the related packets are analysed to individuate the device involved and the change occurred. The *Domotic Agent* then finds the related device in the semantic description that acts consequently. If the events related to the device are also of interest for other Agents, (e.g. for interoperability or for further analysis), the notification of the event is spread properly.

The way that a *Domotic Agent* implements the described functionalities can be different for each domotic system because they depend on many factors like:

- *the media of the bus*: it can be wired like twisted pair, coaxial, ethernet, power line etc. or wireless;
- *the functioning of the domotic system*: what the system permits to do;
- *the available APIs*: what the software that implements the protocol of the domotic system permits to do.

### 4.3 Semantic layer

To exploit their functionalities, including interoperability, each domotic agent exploits three categories of ontologies (Figure 4.1) to work:

- *DomOnt*: is the universal semantic model which abstracts and standardizes the features and functions of the domotic devices independently from their belonging home automation systems. It's shared among all agents;
- *TechOnt*: for each domotic system agent, it provides a semantic model describing the functioning of the domotic system and the available devices conforming to it;
- *TechMapOnt*: for each domotic system agent, a semantic model providing a map between the two previously described ontologies.

The underlying idea is to use *DomOnt* ontology to describe and standardize home smart devices. *DomOnt* is composed by classes of objects organized in an hierarchical tree structure where each node has its own characteristics to which are added those of its ancestors. At startup time, *DomOnt* is only a taxonomy of classes. To be used, *DomOnt* must be filled with the virtual representation of the devices that are in the environment. These virtual devices are represented by ontological *Individuals*. In an ontology, an *Individual* is like an *Instance* in an *Object Oriented programming* language.

The configuration, the available domotic devices and the functioning of each domotic system in the environment, is described using the *TechOnt* ontology. There is a specific *TechOnt* for each specific domotic system. As an example, if in the environment are installed the *KNX* and *UPnP* systems, also *KNXOnt* and *UPnPOnt* ontologies exist that are owned respectively by the *KNX Domotic Agent* and *UPnP Domotic Agent*. Being the *TechOnt* only a description of the domotic system functioning, there is the need of an ontology that is able to put in relation the *DomOnt* with each *TechOnt*. For this reason, for each *TechOnt* a specific *TechMapOnt* is created. These ontology are able to properly map devices stored in the *TechOnt* into *DomOnt*. The *Domotic Agent* acts as a real gateway that interface on one hand, the domotic system and, on the other hand, the abstraction layer of the software platform. For example, let's say that in the domotic system *X* there is a switching lamp that accepts the commands: *setStatus* with a boolean input value *getStatus* with and output boolean value (Figure 3.7). When an interaction between the virtual and physical device is needed to modify the status of the lamp, the corresponding *Domotic Agent*, through the use of the defined ontologies, has to associate the equivalence of the *setSwitch* and *setStatus* functionalities and to associate the equivalence of the input parameter (*on*  $\equiv$  *true*, *off*  $\equiv$  *false*).

Recalling the example of the switch and the lamp bulb, when a button is pressed on the panel, a message that describes the event passes through the *domotic bus*. The *Domotic Agent* captures this message and maps it in the *DomOnt* ontology to make it undersable and usable by the system. In this way, the *Domotic Agent* is able to identify the corresponding ontological *Individual* of the pressed button. The ontological *Individual* contains all the information related to the device and to how it interacts with the other

devices. In this example, the *Individual* is related with the *Individual* of the lamp bulb belonging to another domotic system. The *Domotic Agent* of the button then commands the *Domotic Agent* of the lamp bulb to change the state of that device. In a similar way, the *Domotic Agent* of the lamp bulb receives the command to change the state of the device. The *Domotic Agent* of the lamp bulb locates the right *physical device* that corresponds to the involved *Individual*. According to the information stored in ontologies, the *Domotic Agent* of the lamp bulb generates the message to be sent to the domotic bus to act the change of state of the device.

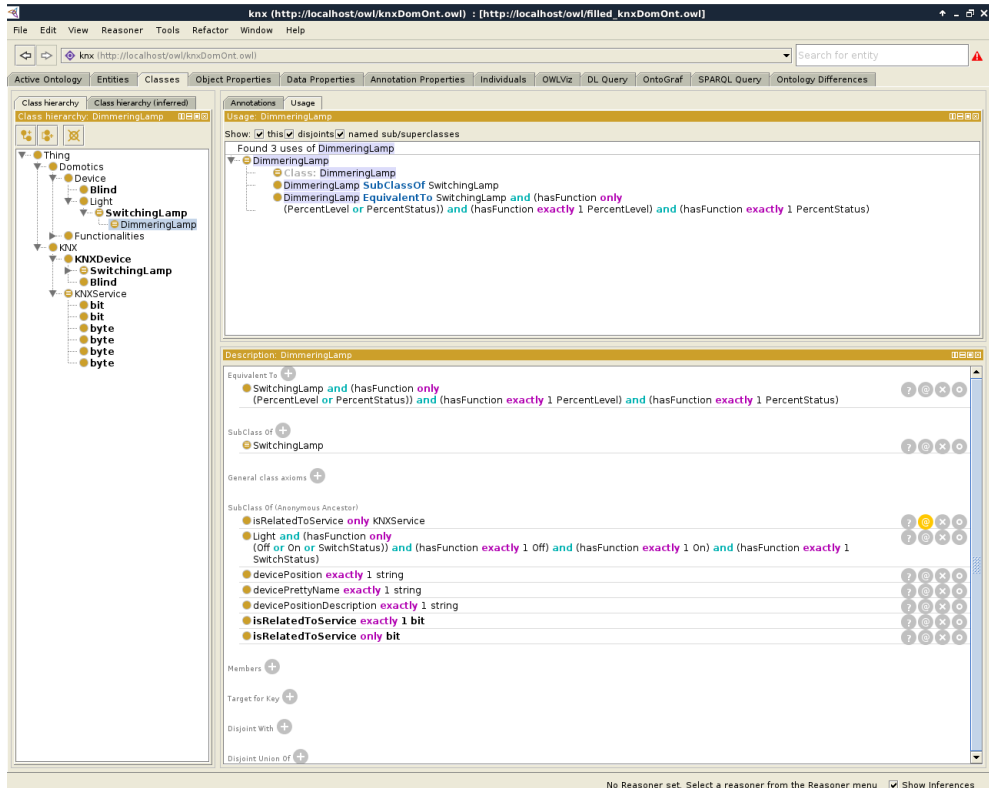
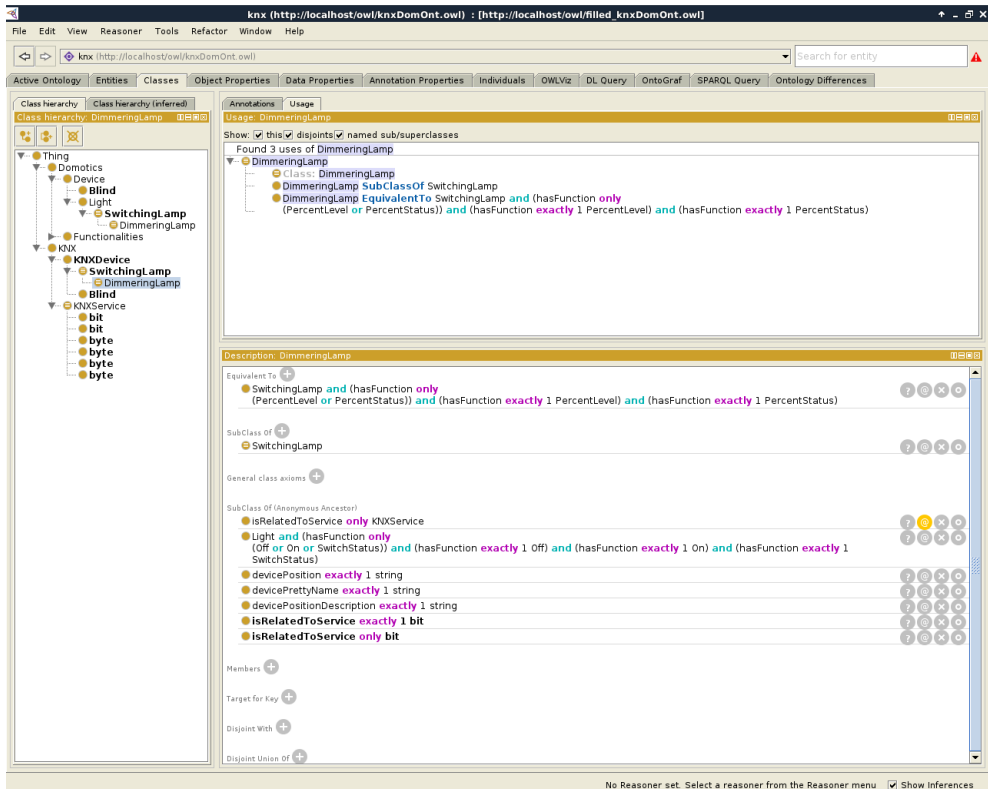


Figure 4.3. A screenshot of the ontology that defines the Dimmering Lamp

In Figure 4.3 is shown an example that defines a *DimmeringLamp* as a *SwitchingLamp* but also with the functionalities to set and get the percentage of light to release. Moreover, in Figure 4.4 are shown the properties for a *KNX device* to be classified as *DimmeringLamp*. In the same ontology there are all the needed information that allow the device to interact with its proper *Domotic bus* for reading and writing operations.





**Figure 4.4.** A screenshot of the ontology that merges the DomOnt with knxDomOnt ontologies for the KNX Domotic Agent

## 4.4 IPv6 and IoT

### 4.4.1 Introduction

Although the Internet has already fundamentally changed society [36], the greatest transformation still lies ahead of us. Several new technologies are now converging in a way that sets the Internet on the brink of a true revolution, as billions of large and small objects are connected and take on their own Web identity.

Following on from the Internet of computers, the next phase of development is the *Internet of Things* [9], when more or less everything will be connected and managed in the virtual world. This transformation will be the Net's largest expansion ever and will have sweeping effects on every industry, and all of our everyday lives.

In fact, in the near future, more and more devices and systems will be capable of sending and receiving data automatically via the Internet. This new scenario involves new developments with enormous potential, for example, in business markets. Indeed, the *Internet of Things* (IoT) will enable connecting market participants and sectors that previously had no business dealings with one another. This will generate new products

and services, which will in turn lead to the creation of new business models as well. In such a scenario, companies must get used to the idea of sitting down and cooperating at a "virtual table". Web-based platforms can create the basis for partners to extend or supplement what they offer in completely new ways. The *Internet of Things* is however not just a distant vision of the future - it's already here and is having an impact on more than just technological developments.

Every device that connects to the Internet requires an *IP* address, and it has been predicted that by the year 2020 there will be 50-100 billion Internet-enabled devices in a world [33] with an expected global population of 7.6 billion people. Considering that, according to a survey of urban environments, each human being is surrounded by 1000 to 5000 trackable objects [86], the move to *IPv6* [47] is necessary, as it provides an almost unimaginable number of *IP addresses* — 18 quintillion blocks of 18 quintillion possible addresses.

Domotics, or home automation in a so-called "Smart Home" involves the controlling and monitoring of home appliances in a unified system. Such control systems include lighting, climate control (HVAC: Heating, Ventilation, and Air Conditioning), security systems and even home electronics. Home automation is closely related to (industrial) building automation, which focuses on the automation of large commercial buildings.

There are many different types of home automation systems available. These systems are typically designed and purchased for different purposes. In fact, one of the major problems in the area is that these different systems are neither interoperable nor interconnected. These systems range from simple remote-controlled light switches to fully integrated, networked devices controlling all appliances in an entire building. Home automation is an extremely appealing application of the *Internet of Things*. It envisions a future home environment where embedded sensors and actuators (e.g., in consumer electronic products and systems) are self-configuring and can be controlled remotely through the Internet, enabling a variety of monitoring and control applications. Such communication capabilities are often offered by manufacturers of domotic systems, though without Internet compatibility. In fact, each manufacturer produces its own special device called an *IP gateway* that enables physically interfacing a proprietary domotic bus with an *IPv4-enabled Ethernet socket*. By connecting the *IP gateway* to the Internet, either directly or through a home / residential gateway [87], the domotic system can be managed remotely using a PC, Smartphone or Tablet through the use of the proper software.

In contrast to the true *IoT* paradigm, such solutions provide the home (but not each device) with a unique Internet access point (and hence, a unique *public IP address* that can be assigned to the *IP gateway* or the home / residential gateway according to the home network configuration) for controlling all the devices connected to a certain domotic bus. In this scenario, the assigned *public IP address* identifies not a single device or function, but the entire domotic network. For this reason, using common Internet applications (e.g. Web browsers) is often not sufficient to interact directly with a single home device through the Internet: in addition to the *IP gateway*, a customized software manager application is required to locate devices and activate their functions within the domotic

network. In this case, the software manager application may be run within the *IP gateway* of the home / residential gateway, but not within the domotic devices themselves, which are lacking a public *IP address*. One example of a software manager application would be a Web server that translates Web browser user interactions into specific domotic commands, and can hence display and enable interaction with the available devices belonging to a certain domotic network.

However, by exploiting the functionalities of these *IP gateways*, it is possible to develop a system for controlling home electrical devices via the Internet using the *IPv6 addressing system* to enable direct control. Thus, the aim of this work consists in developing a software able to interface directly with domotic devices following the principles of the *IoT* paradigm.

#### 4.4.2 State of the art

Current domotic systems differ widely in characteristics, operation, functionality, device management and so on. For these reasons, in general they are not natively interoperable and moreover they use different proprietary media to transmit messages across their own networks. This is because most domotic systems use physically different wired or wireless proprietary buses, and protocols that are incompatible with Internet and *IPv6 technologies*.

To overcome the current lack of direct Internet connectivity, many home automation manufacturers still provide special interfaces called *IP gateways* that interface the domotic system buses with the Internet using the *IPv4 protocol*. As mentioned, this results in the ability to interconnect only the domotic bus and not each single device. Thus, although domotic devices can be piloted remotely through the Internet, it is still impossible to interact directly with them using standard protocols.

Many works have addressed the *IoT paradigm* in some early attempts to make domotic devices compliant with the *IPv6 protocol*.

Two main approaches to assigning *IPv6 addresses* to domotic devices lacking *IPv6-compatible interfaces* have been followed in the literature:

- *hardware*: using a NIC (Network Interface Card) for each domotic device. The NIC acts as a hardware gateway able to interface the device with the network. This solution is poorly scalable because it requires physically modifying and adding hardware to each domotic device;
- *software*: can be implemented without making hardware modifications. It is potentially quick and easy to apply to a virtually infinite number of devices without additional single hardware costs.

Tin-Yu Wu et al. [90] has proposed an interesting approach to intelligent appliance auto-configuration in a networked domain. It implements three functions: (i) assisting the information appliance in acquiring a regular domain name without manual configuration; (ii) providing a session initialization protocol, uniform resource identifier, auto-configuration

and seeing to device registration; (iii) initiating communications between devices in order to manage the residential gateway and configure the user management system interface. Unfortunately, this solution requires adding a software application that must be executed during the device system boot, which involves modifying the original device design by adding different embedded routines.

Another work in this regard has been presented by van Moergestel and Meyer [84], who propose a multi-agent based architecture to implement home automation device interoperability. The architecture described provides each device with an independent agent to which an *IPv6* address can then be assigned. However, the solution requires installing small computer equipment to execute the corresponding agent on each device, thus making the system poorly scalable.

Jung et al. [50] propose an approach using a gateway that allows the integration of a building's automation system into constrained *RESTful environments* by means of a per-device *oBIX-based IPv6 interface*. This solution does not require any direct manipulation or adjustment of the devices. In another work, Jung et al. [49] propose a solution for adding *IPv6* and illustrate it using the *BACNet system* as an example. In the example, however, significant modifications are made to the *BACNet protocol* functioning.

Jeong et al. [48] also address the problem of *IPv6* and the *Internet of Things*, proposing a solution that permits users to discover, identify and communicate with things independently of their underlying addresses and network protocol stacks. The solution exploits the *EPC (Electronic Product Code)*, a worldwide standard that provides a unique identity for every physical object anywhere in the world via *RFID* or optical data carriers. The solution proposed does not modify either standard protocols or devices, but implements an ad hoc network infrastructure.

#### 4.4.3 The *IPv6* gateway

Exploiting the peculiar feature of the *software platform*, that is, its ability to provide a uniform system view of networked devices implementing different technologies, the aim of this gateway is to develop a module to make domotic devices compliant with *IoT* goals. The gateway is perfectly scalable and immediately adoptable without the need for hardware or hybrid solutions that may require direct device alterations, which can in general be risky and difficult for non-specialized users. Moreover, modifying the hardware of a large number of devices can be very demanding, especially when some are not physically accessible.

The proposed solution aims to:

- link together each *physical device* via an *IPv6 address* using a mapping scheme. This function creates a correspondence between the ontological *URI* of the *virtual device* and the assigned *IPv6 address* enabling a translation function that permits to find the address from one representation to an other bi-directionally;

- make each *physical device* managed by the *software system* reachable anywhere and any way through an *IPv6 address*. Obviously, all *Pv6 addresses* associated to domotic devices must be globally valid, unique and fully compliant with *IPv6 system*;
- implement a dedicated *Model View Controller (MVC)* [71] Web interface for each *physical device*. This interface, which must be reachable using the device's *IPv6 address*, displays each device's status and available functions.

The solution requires assigning a set of *global IPv6 addresses* to each machine running a *Domotic Agent*. Each *IP address* in the set, created by the *Domotic Agent* machine itself, must include two main fields in order to be valid:

- a prefix that it is used for routing purposes. This can be achieved by exploiting *OS functions* and *APIs*. On a Linux machine, for instance, the `ip` command line is invoked to obtain the network prefix field and assign an address to the network interface;
- an interface identifier, which distinguishes the host network interface. This can be created randomly by the server machine itself. If combining the prefix with the interface identifier leads to the generation of an already existing *IPv6 address*, a *Duplicate Address Detection* error of the *Address Resolution Protocol (NDP)* is reported.

When a *Domotic Agent* notifies a new *physical device* in its network, it creates the *virtual device's* representation and it associates to it an *IPv6 address* providing dual identification. In doing so, the *Domotic Agent* compiles a bi-directional map in order to enable identification of address correspondences. When an *IPv6 request* is forwarded by a resource belonging to the *Internet of Things*, the *Domotic Agent* responsible for the requester's address responds. The mapping scheme is then used to find the correspondence between the *IPv6 address* and the *ontological physical device URI* to identify the correct device involved. To process the request, the *Domotic Agent* translates the request to a formalism (the proper domotic system language) that the device can process and queries the device involved. The device's response is delivered to the *Domotic Agent* which finally sends it to the requester.

A dynamic *Domain Name server (DNS)* service cooperates with the *IPv6 gateway* and the *Domotic Agent*. In this way, accessing a network resource does not require specifying long, difficult *IPv6 addresses*. A *Bind9 DNS* server [64] is used for this purpose. In order to automatically and dynamically assign names to the entries in the DNS database, each time a new *virtual device* joins the *Domotic Agent* network and requests a new *IPv6 address*, a *DNS entry* is created using the device name.

#### 4.4.4 The web interface

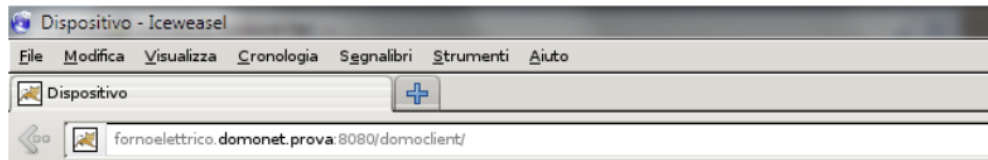
One of the main objectives of the *Internet of Things* is to enable direct interaction with devices. A simple way to achieve this is through a Web interface. However, unfortunately domotic devices cannot in general interact directly with Web browsers because they lack standard Internet protocols, such as *HTTP*, *ldap*, *imap*, *pop*, and so on. Moreover, implementing such protocols directly on domotic devices is not possible due to their hardware limitations. For these reasons, the Internet protocols must be implemented elsewhere.

The implementation includes the use of a *Web server* running on the same computer where a *Domotic Agent* is running. This *Web server* can respond to *http requests* as if they were direct Web interactions with a device identified by its *IPv6 address*. As test, the *Tomcat Web server* was used to display all the necessary *Web pages*.

Due to the potentially extremely different characteristics of the various devices in the network, in order to avoid generating many distinct *Web pages* for each device, the *Web server* must rely on a dynamic page generator. To this end, *JavaBean*, *JSP* and *Servlet* technologies [6] have been adopted in the test and applied under the constraints of the MVC [6] paradigm. In this way, the *Web server* responds to requests with *Web interfaces* customized to each device, which are invoked using the device's *IPv6 address* directly. When a Web request originates from the Internet, *Tomcat Web server* responds because it shares the same set of *IPv6 addresses* as the *Domotic Agent*. The request is received by a *servlet*, which reads the address of the device involved in the communication from the requester URL. Using the *IPv6 - ontology URI* mapping, the *Domotic Agent* then identifies the device and queries it in order to obtain the information needed to satisfy the request. To display the results on a *Web page*, the *Domotic Agent* creates a *JavaBean* to represent the device involved and fills it with the needed data. *Tomcat* takes the completed *JavaBean* as input and shows the result through a *JSP*.

To test the platform's functioning, some test *Web pages* for controlling a device have been created.

Using a Web browser, a request was sent to load the main page of the device, which in this example, was an electric oven (Figure 4.5).



## Electric Oven

### Available functions

[Electric Oven State](#)

[Electric Oven Socket State](#)

[On/Off Electric Oven](#)

[Enable/Disable Electric Oven Socket](#)

**Figure 4.5.** Device callable functions

This resulting page shows the functions that can be activated, in particular: request the device state, power it on or off, check the state of the electrical socket and enable or

disable it. For instance, the oven power on or off function can be activated by specifying the needed parameters (Figure 4.6).



Servizio - Iceweasel

File Modifica Visualizza Cronologia Segnalibri Strumenti Aiuto

Servizio

fornoelettrico.domonet.prova:8080/domoclient/execute?sid=1

## Electric Oven

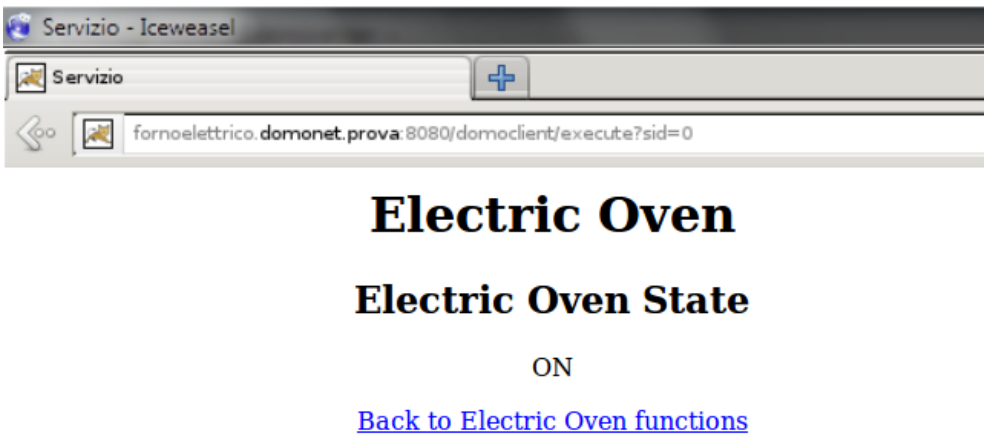
### On/Off Electric Oven

New Electric Oven State:

Submit

**Figure 4.6.** Device function call

When the function has been activated, the response shows the updated state of the oven (Figure 4.7).



Servizio - Iceweasel

Servizio

fornoelettrico.domonet.prova:8080/domoclient/execute?sid=0

## Electric Oven

### Electric Oven State

ON

[Back to Electric Oven functions](#)

**Figure 4.7.** Response of the device function call





## User agent

### 5.1 Introduction

The final users of the *Software Platform* are humans that live in an environment where the system is installed and running. One of the principal objectives of today's *Ambient Intelligence* environments is to provide users with services according to their activity, i.e. preferences in order to accomplish a specific user's goal. In fact, each user has different needs and habits and it is not possible to generalize them using an unique universal model for all of them. For this purpose, miscellaneous user information must be collected and structured into user profiles. These profiles offer the advantage of being easily enriched and exploitable by the environment, in order to deliver to the user, at any moment and at any place, the best fitted service, with regard to his activity.

The *User Agents* are the digital representation of users that interact with the system and are able to collect and structure user's profiles.

### 5.2 Semantic layer

Each *User Agent* contains an ontology that profiles a specific user. The type of information that is stored in the ontology depends and varies according to the specific applications implemented by the *Software Platform*. As an example, if the aim is to analyse a clinical situation of a user, the ontology may contain data related to medical parameters. For convenience, instead, the ontology may contain data related to preferred home settings.

In Figure 5.1 and 5.1 are shown parts of the structures of the implemented ontology that are needed for the developed *Software Platform*. These structures cover:

- biological information like blood pressure, body temperature;
- preferences like brightness level, humidity, noise level, temperature and so on;
- person related information like disabilities, dislikes, physical information and so on.

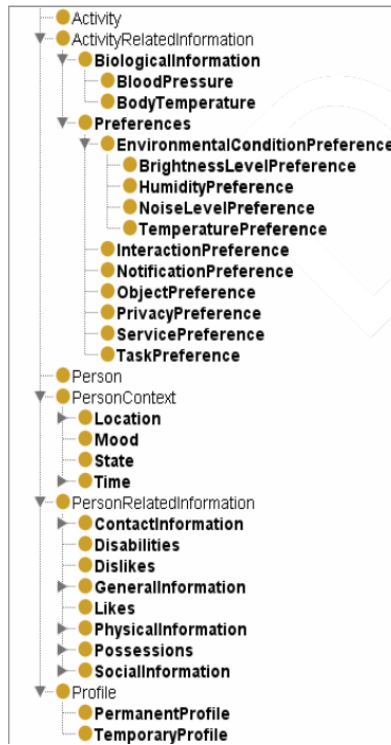


Figure 5.1. Class hierarchy of the ontology

### 5.3 Track and identification of users

The tracking and identification of users is a challenging problem that is related mainly to the complexity of the smart home environmental conditions and the variety of services and users' demands. The literature proposes different approaches. In general, these approaches can be classified into three major categories that adopt a different concept, strategy, and design to deal with the problem:

- tagged identification and tracking including Infrared, Ultrasonic, and Radio frequency based approaches;
- non-tagged identification and tracking including machine vision, smart floor, and wireless distributed PIR approaches;
- multimodal identification and tracking including smart floor with RFID, smart floor with machine vision, and machine vision with laser scanners approaches.

However, these solutions often fail because they don't take into account the problems related to privacy and user's preferences like in the case of videocameras use. Other approaches are still inapplicable due to several fundamental problems. For instance, the majority of smart floor approaches suffer from unrealistic overall cost and system hardware complexity. Considerable research works have been directed toward multimodality

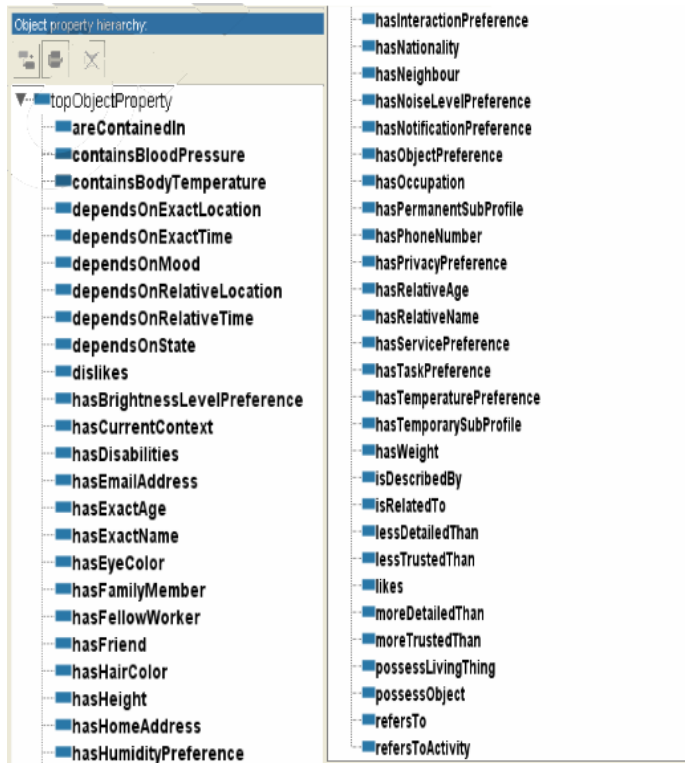


Figure 5.2. List of properties of the ontology

identification and tracking approaches as a way to enhance system performance through strengthening complementation and overcoming or minimizing their weaknesses. However, these approaches often have their own problems and limitations, such as high costs, due to the improper combination of strategies utilized.

The majority of services related to *Ambient Intelligence* are context-aware and/or person-aware. This means that there is the need to provide appropriate services to the right person at the right place and at the right time. Therefore, the human identification and tracking (HIT) approach should be able to support all context-aware services by fulfilling the following requirements:

- context-aware applications demand the information of person's location to provide their services efficiently. Less than 0.3 m error range of person's location is usually required. Thus, the HIT approach should be able to detect person's presence and determine their location accurately;
- recognising the residents' identities is essential to ensure providing the appropriate services to the right person. Some personal-aware services, specifically health care, require reliable identification while dealing with older and/or disadvantage people. Therefore, the HIT approach should be able to provide great accuracy in personal identification, i.e. the CCR should be greater than 98%;

- reliability and robustness are both vital for the HIT approach because person-aware services are dealing with humans, and mistakes could be too expensive to pay. In addition, low cost for mass market, scalable to satisfy various environments, low complexity for ease of use and maintenance, ambient for minimum intrusion and privacy are required as well;
- the HIT approach should detect, track, and identify inhabitants automatically without imposing particular conditions like the need of wearing or carrying a special device.

In this work, the recognition of the user who performs the actions is still an open issue unless the interaction with the environment is acted by a computer interface requiring identification. For this reason, for testing purposes, the environment is considered inhabited by one person.

## 5.4 User interaction with the environment

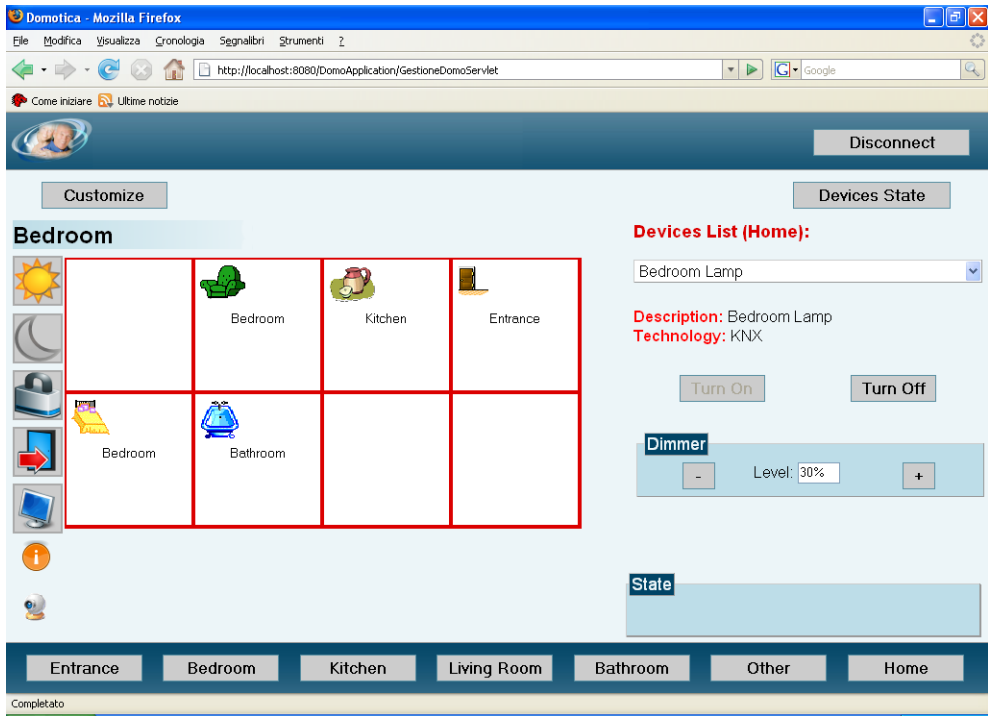
As shown in Figure 3.3, the user can interact with the environment in two main ways: acting physically with the device (e.g. acting on the light switch) or using a remote system (e.g. a web interface or a remote control).

As shown in section 5.3, the physical interaction with devices poses some problems related to the identification of the user in order to track his actions. On the contrary, acting through a remote interface, like a PC, a tablet, a smart-phone and so on, it is possible to implement authentication methods that are useful to recognize users.

Combining the ontological abstraction layer of domotic devices created in *Domotic Agents* (that permits to obtain an uniform and formalized description of devices and their functionalities regardless of their belonging technology) with the information stored in the user's profile, it is possible to build at run time any type of *Human Machine Interface* (HMI) interface tailored on user's abilities.

In this work, exploiting the same ontological device descriptions, three different interfaces are implemented for three different control devices:

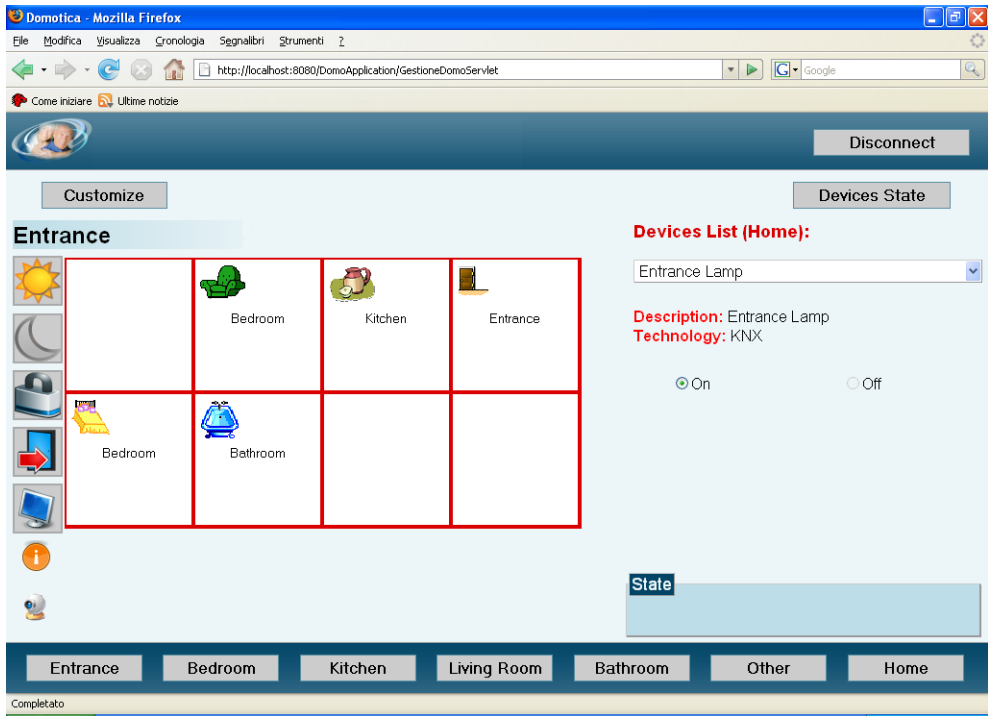
- *web interface* (Figure 5.3 and Figure 5.4): it is possible to interact with the domotic devices using a common browser for PC. The application is implemented using *Tomcat Web Server*, *Servlet*, and *JSP* exploiting the *MVC* paradigm. The user interface is composed by three grouping elements: the header (containing a logo/title), the main area, and the footer. In the footer are represented the rooms available in the home and some general controls. In the main area there is a map zone (which provides a graphical representation of the rooms available), a device area with a list of the available devices and the controls for the currently selected device. The user's interface is build dynamically: depending on the type of device selected at run time, different controls will be shown in the user interface in order to operate with it. This has been obtained by exploiting the device abstraction.
- *Android app* (Figure 5.5): an interface implemented as App for smartphone and tablet based on Android OS. It provides *Text To Speech* (TTS) and *Automatic Speech*



**Figure 5.3.** Web interface used to control the home environment: light

*Recognition (ASR)* features. The starting point for the creation of the mobile version is still the same as for the web interface. In this case, smaller icons and, generally, more simplified representations are used. For a more suitable visualization on small screens, the splitting of the user interface in areas depends on the information regarding the screen size of the current device.

- *remote control* (Figure 5.6): *Philips PRONTO TSU 9400* is a programmable specialized device that works like an advanced traditional remote control for TV. This device is provided with wifi and infrared connections. For the creation of an intuitive and user friendly interface, three characteristics were identified: (i) an interface that would allow the use of a wifi socket to communicate with the system; (ii) a compact dimension of the device but with a large monitor that allows easy viewing of the menu and a rapid management of devices in the home; (iii) usable in a simple manner from any type of person (especially elderly). In addition, the user interface is familiar as it has the same characteristics of a device present in all homes. Because of the limited hardware resources of the device, you have created an intermediary gateway between the remote control and the *Software Platform* that implements the same functionalities to support device operations.



**Figure 5.4.** Web interface used to control the home environment: light with radio buttons



**Figure 5.5.** Interface for Android used to control the home environment

Moreover, if the user suffers from a disability, discovered by the user's profile (for example he has visual deficits) the same interfaces can be automatically adapted as needed (for example, increasing standard fonts and using higher contrast of colours).



Figure 5.6. The "Pronto" remote control





## Intelligent agent

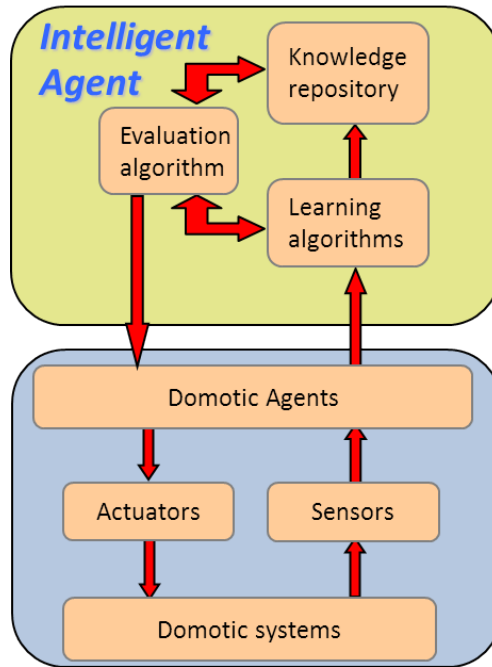
### 6.1 Introduction

Intelligent agents are special entities of the *software platform* that are able to interact actively with the life of users. *Intelligent agents* are able to participate pro-actively in users' life interacting directly and autonomously with the environment to support and offer services to users. To do that, the agents are usually equipped with special algorithms of artificial intelligence and they are able to query the knowledge bases of *domotic agents* and *user agents*.

Exploiting an Artificial Intelligence system based on an *Aml* paradigm, the system configuration stage of *Intelligent agent* is limited to the physical installation of the devices and software without regard to adjustments and settings, which often can be difficult to understand and put into practice. Users can simply go about their usual business within their living quarters and just ignore the technology surrounding them.

The life-cycle of an *Intelligent agent* can be divided into 4 steps (Figure 6.1):

- *the information collection step* is activated when a device changes state and consequently a multicast update message is sent to all *Intelligent agents* that have subscribed to the update notification service, according to the publish/subscribe design pattern paradigm. At the same time, a log manager archives the information about the updates in an *XML* file for diagnostic purposes and appropriate analysis. It stores all the actions performed by users, the device that has changed status, the new status and the identifier of the user who performed the action, as well as the time of activation;
- *the collected information analysis step* identifies the sets of actions that may lead to the recognition and creation of new scenarios;
- if the analysis step has recognized a new scenario to be created, *the creation / removal of rules step* will create new rules that represent the application of the new scenario. Previously learned rules are modified by the system when, over time, a change in learned habits or external factors occurs. In such circumstances, the system is



**Figure 6.1.** Intelligent agent architecture

able to remove and eventually substitute previously learned rules that are deemed no longer valid according to newly acquired experience;

- finally, during *the rules execution step*, the prerequisites for a learned rule are verified, the rule is applied by the system by invoking the corresponding commands to be routed to the appropriate devices.

In pervasive environments, a *scenario* is defined as a set of user's actions that are in some way related to each other. A *scenario* represents a particular configuration of some devices in a particular moment. The particular moment can be, for example, the reaching of a certain time of the day or the execution of specific actions by the user. A *scenario* is represented digitally through a rule that formally describes the specific configurations of the devices for that particular moment. Moreover, the *context* is defined as the representation of the knowledge that a system has about its own state. The context is not simply a snapshot of the environment at a particular time, but instead it usually represents information over a given period of time during the life of the environment itself. The context provides the knowledge bases for systems that learn from past *contexts* and experiences, thereby allowing for advanced adaptive capacities and facilitating proactive decision support with varying degrees of autonomy.

In this chapter, are presented two types of *Intelligent agents*. The first has the objective to learn user's habits and to anticipate them automatically; the second, has the objective to anticipate possible dangerous situations for user's health.

### 6.1.1 Associative rules

The methodology known as *association analysis* is useful for discovering interesting relationships hidden in large data sets. The uncovered relationships, that have to be extracted from the know data, can be represented in the form of *association rules* or *sets of frequent items*. An association rule is an implication expression of the form  $X \Rightarrow Y$ , where  $X$  and  $Y$  are disjoint itemsets. The strength of an association rule can be measured in terms of its support and confidence. *Support* determines how often a rule is applicable to a given data set:

$$\text{Support}, s(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{N} \quad (6.1)$$

while confidence determines how frequently items in  $Y$  appear in transactions that contain  $X$ :

$$\text{Confidence}, c(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)} \quad (6.2)$$

*Support* is an important measure because a rule that has very low support may occur simply by chance. A low support rule is also likely to be uninteresting from a business perspective because it may not be profitable to promote items that customers seldomly buy together. For these reasons, support is often used to eliminate uninteresting rules. *Support* also has a desirable property that can be exploited for the efficient discovery of association rules. *Confidence*, on the other hand, measures the reliability of the inference made by a rule. For a given rule  $X \Rightarrow Y$ , the higher the confidence, the more likely it is for  $Y$  to be present in transactions that contain  $X$ . Confidence also provides an estimate of the conditional probability of  $Y$  given  $X$ .

*Association analysis* results should be interpreted with caution. The inference made by an *Association Rule* does not necessarily imply causality. Instead, it suggests a strong co-occurrence relationship between items in the antecedent and consequent of the rule. Causality, on the other hand, requires knowledge about the causal and effect attributes in the data and typically involves relationships occurring over time (e.g., ozone depletion leads to global warming).

A common strategy adopted by many *Association Rule* mining algorithms is to decompose the problem into two major subtasks:

- *Frequent Itemset Generation*: whose objective is to find all the itemsets that satisfy the `minsup` threshold. These itemsets are called frequent itemsets;
- *Rule Generation*: whose objective is to extract all the high-confidence rules from the frequent itemsets found in the previous step. These rules are called strong rules.

The computational requirements for frequent itemset generation are generally more expensive than those of rule generation.

#### Frequent Itemset Generation: Apriori

The *Apriori* algorithm is an influential algorithm for mining frequent itemsets for boolean *Association Rules*. It uses a "bottom up" approach, where frequent subsets are extended

one item at a time (a step known as candidate generation, and groups of candidates are tested against the data. *Apriori* is designed to operate on database containing transactions (for example, collections of items bought by customers, or details of a website frequentation).

The steps to perform the *Apriori* algorithm are:

1. Let  $k = 1$
2. Generate frequent *itemsets* of length 1
3. Repeat until no new frequent itemset is identified
  - a) Generate length  $(k+1)$  candidate itemsets from length  $k$  frequent itemsets
  - b) Prune candidate itemsets containing subsets of length  $k$  that are infrequent (How many  $k$ -itemsets contained in a  $(k+1)$ -itemset?)
  - c) Count the support of each candidate by scanning the DB
  - d) Eliminate candidates that are infrequent, leaving only those that are frequent

Using the  $F_{k-1} \times F_{k-1}$  Method, the candidate generation procedure in the *Apriori* merges a pair of frequent  $(k - 1)$ -itemsets only if their first  $k - 2$  items are identical. Let  $A = a_1, a_2, \dots, a_{k-1}$  and  $B = b_1, b_2, \dots, b_{k-1}$  be a pair of frequent  $(k - 1)$ -itemsets. A and B are merged if they satisfy the following conditions:

$$a_i = b_i \text{ (for } i = 1, 2, \dots, k - 2) \quad (6.3)$$

and

$$a_{k-1} \neq b_{k-1} \quad (6.4)$$

## 6.2 Agent that learns and anticipates user needs

By monitoring users' activities in a highly enriched domotic home environment, it is possible to learn to recognize such *scenarios* and anticipate the needs of its inhabitants.

Given the wide variety of different scenarios that may arise, the agent is made up of two complementary, interoperating modules: *the association* and the *statistical rules* managers (Figure 6.2).

Working together, they perform real-time analyses aimed at discerning sequences of events, that is, a set of interactions with the domotic environment that occur within a fixed short time span (a few seconds, minutes or hours), though not necessarily in a specific sequence, and may therefore represent user's habits.

### 6.2.1 Association rules manager

The association rules manager is responsible for learning scenarios made up of a set of actions habitually carried out by the user. These scenarios are made up of actions related to each other in the sense that they occur within a short interval from each other, but are unrelated to any specific time of execution. These are called non-temporal scenarios.

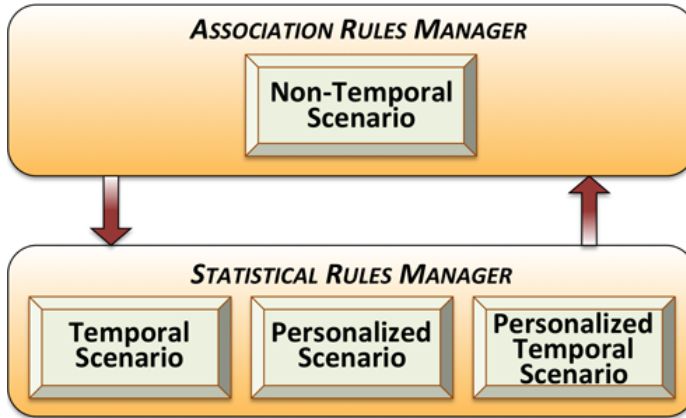


Figure 6.2. Modules of the Intelligent agent

Because it is very likely that some actions are preset in most scenarios, the necessary condition for determining a *non-temporal scenario* is that a user perform a set of actions sufficiently regularly to discriminate the scenario to be actuated. For example, a user who, leaving the house, habitually turns off all the lights and closes all the shutters, may occasionally also shut off the gas valve and turn off the TV or stereo, or even forget a light on or leave the shutters deliberately open so that the sunshine heats the house in the winter.

```

private void aprioriGenRules(Itemset frequent, LinkedList<Itemset> consequents) {
    LinkedList<Itemset> ok = new LinkedList<Itemset>();
    if ((consequents.size() > 0) && (frequent.length() > consequents.get(0).length())) {
        for (int i = 0; i < consequents.size(); i++) {
            Itemset predecessor = new Itemset(frequent.getItemsetCode());
            predecessor.remove(consequents.get(i).getItemsetCode());
            if (!blackList.contains(predecessor, consequents.get(i))) {
                float conf =
                    ((float) itemsetSupportCount.get(frequent)) / itemsetSupportCount.get(predecessor);
                if (conf >= minConfPerc) {
                    HashSet<Itemset> temp = rules.get(predecessor);
                    if (temp != null) temp.add(consequents.get(i));
                    else {
                        temp = new HashSet<Itemset>();
                        temp.add(consequents.get(i));
                    }
                    rules.put(predecessor, temp);
                    ok.add(consequents.get(i));
                }
            } else
                System.out.println("Rule: " + predecessor.getItemsetCode() + " -> " +
                    consequents.get(i).getItemsetCode() + " not considered because it is in the blacklist. ");
        }
    }
    if (ok.size() > 1) {
        LinkedList<Itemset> newConsequents = aprioriGen(ok, ok.get(0).length());
        aprioriGenRules(frequent, newConsequents);
    }
}
  
```

Figure 6.3. The Apriori algorithm

In order to recognize such patterns, the agent applies the *Data Mining* paradigm, in particular, the association rules method. This allows the generation of opportune rules using binary partitions of the itemset that determine a scenario being learned. It is thus possible to anticipate upcoming actions when a user performs antecedents of such actions (i.e. actions leading up to them).

In order to generate the frequent item-sets (which are the potential scenarios), the constraints of the *Apriori* algorithm are used (Figure 6.3), which generate candidate action sequences via the standard method defined as  $F_{k-1} \times F_{k-1}$  [79].

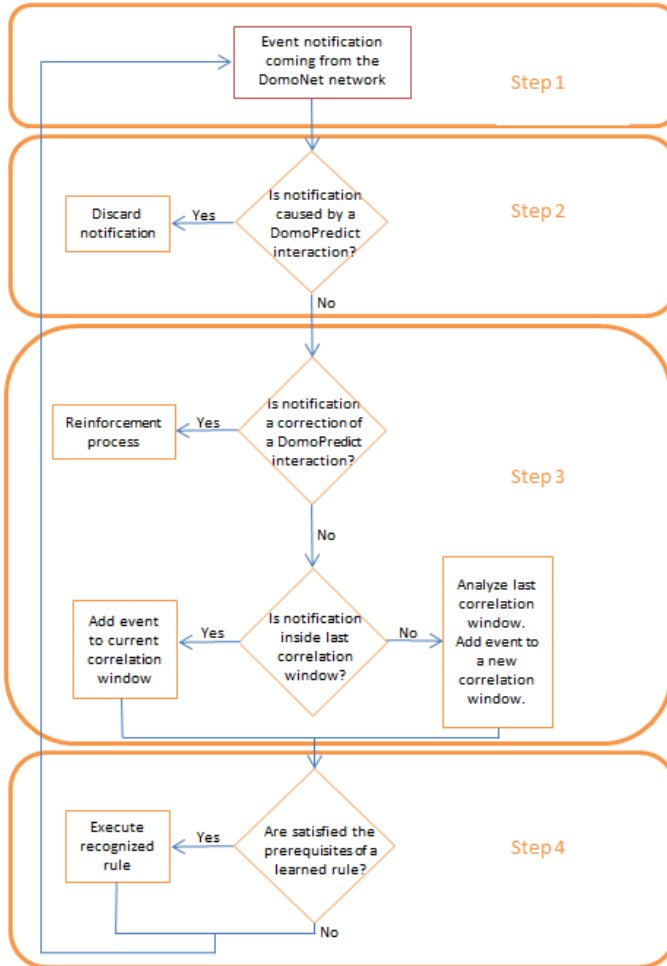
The rule for discovering whether a specific scenario is to be activated automatically is represented via the form  $X \Rightarrow Y$ , where  $X$  and  $Y$  represent the two sets of events into which each scenario is divided.  $X$  contains the precondition events for scenario activation, while  $Y$  contains all other remaining events. The result is: if  $X$  occurs, then  $Y$  is also performed. The strength of an associative rule can be measured as a function of its *support* and its *confidence*. *Support* represents the fraction of scenarios that contain both  $X$  and  $Y$ , while *confidence* instead represents how often the events in  $Y$  are also present in a scenario containing  $X$ .

The problem of discovering *Associative Rules*, given a set of scenarios, consists in finding all the rules whose *support* and *confidence* values are equal to or greater than pre-set thresholds. The size of the dataset is very important for the proper functioning of data mining algorithms. Indeed, data sets can be very large, often requiring days of computer time to create a single model. From this prospective, usual *Data Mining* is unsuitable for these purposes, because the dataset is empty at start-up and is created in real time. The solution found to overcome this obstacle is to act on the *support* parameter of the *Apriori* algorithm, bearing in mind that the few data available initially could lead to the acquisition of erroneous habits. In order to limit such erroneous habits acquisition, the dataset is enriched with a new item-set only when the minimum support parameter is greater than a prefixed threshold. As this parameter is used to evaluate whether or not a group of actions is frequent, simply increasing its value when dealing with small datasets will make it more difficult for a given scenario to be learned, thereby preventing infrequent item-sets from being considered. Thus, at first, a rather high value is set, which is then decreased in the long term proportionally with the increasing population size of the dataset, so that eventually most item-sets will be deemed frequent, thereby allowing even rather rare habits to be learned.

The choice of implementing non-temporal scenarios using associative rules was based on the consideration that sufficient training data is unavailable at start-up time and the dimensions of the learning system hypothesis space should not be fixed. Thus, an *unsupervised machine learning* system had to be adopted, in order to allow the creation of relationships and groupings between similar data. Moreover, the possible scenario to enact must be chosen among a finite set of possibilities.

To create a new non-temporal scenario rule, a method is needed that can efficiently perform the following two tasks: (i) classify a group of actions as a scenario and (ii) find a way to enact this scenario after learning it. Indeed, only in ideal cases would users per-

form the exact same sequence constituting a scenario. Scenarios are much more likely to be identified through a subset of the entire action sequence making up the scenario.



**Figure 6.4.** Association rules manager life-cycle

The first task (Figure 6.4) that must be performed in this process is to arrive at some groupings of potentially correlated actions, which will be represented by those performed in succession over a brief time interval. Such time interval, called the *correlation window*, is a system parameter whose duration can be adjusted as needed. It does not represent an absolute time interval within which actions must be performed in order to be deemed correlated, but it is the maximum time gap that indicates a logical dependency between the last action performed by the user and the previously obtained set of correlated actions. Such procedure necessarily requires a stage of data preprocessing, so as to determine

the minimum set of actions that enables recognizing the correct scenario. The result is a table containing all the action sequences performed over a brief interval and therefore potentially correlated. This is called a sequence table and will be used as the basis for the subsequent stage of learning the scenarios.

The second task is performed under the constraint that any action belonging to a specific scenario cannot be allocated to any other scenario.

It is worthwhile underlining that the efficiency of the algorithm does not depend on the order of the completed actions, but only on the temporal correlations existing between the actions performed, since the actions making up a scenario are not always performed in the same order. Moreover, acquired scenarios are continually subject to modification. An essential requirement for software efficiency is the ability to quickly adapt to users' habits.

This goal is achieved by implementing a reinforcement function (Figure 6.5) in the machine learning procedure. This function is activated by users when they unconsciously correct undesired or incorrect system actions. For example, if the system enters a scenario that calls for switching on a light, and the user switches off that very light, this fact permits the system to "understand" that this particular scenario is incorrect. Another example is when users modify their habits with changing seasons. The system adapts itself automatically to new user habits (e.g. turning off the lights later or opening the windows more often in springtime), without changing any hardware device settings. At the beginning, the presence of this reinforcement procedure is essential in order to enable the system to modify any erroneously learned rules due to a dearth of collected experience. The erroneous rules are relegated to a blacklist and cannot be re-learned for a period of time proportional to the number of times the user has provided negative feedback.

For the activation of non-temporal scenarios it is necessary to identify the minimum set of actions that enables identifying the scenario to be applied. Automatic identification of this minimum set is the most important and critical feature of the system. For example, let us suppose that the system has learned a scenario that includes the two rules: switch on the light in the living room and switch on the TV. Once the user has switched the light on in the living room, it must be determined whether (s)he wants to turn on the TV as well. To do this, it is necessary to calculate the probability that the performance of a group of actions belonging to a scenario implies execution of the others in that same scenario. The conditional relation doesn't need to be one of certainty - a high probability is a sufficient ground for anticipating the need. The associative rules method permits to create a rule in such a way that it associates groups of antecedent actions for executing groups of consequent actions.

The removal of a scenario comes about when the confidence level does not, within a certain time period, reach the pre-set threshold value for triggering the scenario.

### 6.2.2 Statistical rules manager

The *statistical rules manager* is designed to learn scenarios that are not captured by the *association rules manager*. These scenarios include events occurring with systematic



```

private boolean checkReinforcement(DomoMessage domoMessage, Calendar currentTime) {
    boolean result = false;
    long diff =
        currentTime.getTimeInMillis() - executedActiveItemset.getStartTIme().getTimeInMillis();
    if (diff < reinforcementWindow) {
        String removed = checkAndRemoveIfExecutedContains(domoMessage);
        Itemset toBeRemoved = new Itemset(removed);
        if (removed != null) {
            System.out.println("Removing precondition actions: " +
                executedActiveItemset.getItemsetCode());
            HashSet<Itemset> oldRules = rules.remove(executedActiveItemset);
            Iterator<Itemset> iter = oldRules.iterator();
            HashSet<Itemset> newRules = new HashSet<Itemset>();
            while (iter.hasNext()) {
                Itemset current = iter.next();
                System.out.println("Removing postcondition actions: " +
                    current.getItemsetCode());
                if (current.contains(toBeRemoved)) {
                    Itemset newCurrent = new Itemset(current.getItemsetCode());
                    newCurrent.remove(removed);
                    if (newCurrent.length() > 0) {
                        System.out.println("Modified rule: " + executedActiveItemset.getItemsetCode() +
                            " -> " + current.getItemsetCode());
                        newRules.add(newCurrent);
                    } else {
                        System.out.println("Removed rule");
                    }
                } else {
                    System.out.println("Adding rule");
                    newRules.add(current);
                }
            }
            System.out.println("Rule added to black list: " +
                executedActiveItemset.getItemsetCode() + " -> " + toBeRemoved.getItemsetCode());
            blackList.add(executedActiveItemset, toBeRemoved);
            if (newRules.size() != 0)
                rules.put(executedActiveItemset, newRules);
            else {
                System.out.println("No rules left");
                executedActiveItemset = null;
                executedConsequents = null;
            }
            result = true;
        }
    } else {
        executedActiveItemset = null;
        executedConsequents = null;
    }
    return result;
}

```

Figure 6.5. Reinforcement algorithm

periodicity (i.e. each day at the same time or during different periods of the year), as well as the user's living quarters preferences.

To this end, the module creates a user profile obtained by statistically analysing the frequency and percentage of use of appliances. Activation of such preferences can be tied to the particular moment or a sequence of events that has occurred.

Percentages of use are calculated both on a daily basis and for shorter periods of time (e.g. in the morning from 8 to 9 am). This enables identifying potential habitual system states that may present at certain times of the day, without the user performing a sequence of actions.

Collected data are recorded in structures called *UsageTables* in the pair format `<device state, percentage>`, which indicates either the percentage of time a device is in a particular state, or the percentage time that certain events occur (for instance, listening to favourite music or maintaining a room temperature). A number of different time-frames (daily, weekly, seasonal and perpetual) are considered and a different *UsageTable* is created for each.

Such data are used to satisfy user's preferences through a conditional rule of the sort `condition => set preferences`, where the `condition` is dictated by specific events, such as for example, waking up, returning home, a new season's start, and so forth.

The scenarios captured by the statistical rules manager are:

- *temporal scenarios*: these are made up of one or more events usually occurring at the same time of day or for a long period of time. Once the constituent events of the scenario are learned, they are executed automatically at the pre-established time. To build temporal scenarios it is sufficient to observe the relations between actions and time. A reasonable choice seems to be to observe the activation time of the scenario within a pre-set number of days prior. If an action is performed every day at a certain time within this pre-set *time window*, we can assume that a relation exists between the action taken and the time of the day it is taken, and we can thus have it performed automatically. The time for its execution can be calculated by taking the average time at which the action was initiated over the preceding days. The removal of a temporal scenario is accomplished exclusively through a negative reinforcement mechanism, that is, when a user performs an action contrary to that learned by the system;
- *personalized scenarios*: these define a set of actions/parameters that the system uses to configure the environment according to the inhabitant's personal preferences. The learning of a personalized scenario aims to increase the comfort and safety of the user within the environment. To this end, the user's preferences learned over time through his/her daily device usage are analysed. The system is preconfigured to learn the temperature, lighting levels, favourite musical genre and the values of the inhabitant's main vital functions, so as to allow for constant monitoring of the state of health. The rules are created statistically based on the normal distribution over time of the parameter values automatically learned by the system. Activation of a scenario depends on the occurrence of certain situations, such as for instance when a user enters the living quarters. In this case, the actions taken are aimed at controlling the environment according to the user's preferences learned;
- *personalized temporal scenarios*: these represent the living parameters learned by analysing any customary preferences repeatedly set by the user at specific times (e.g., which take place each day or each week at the same time). Although learning this type of scenario depends on user execution at specific times, its removal is accomplished in the same way as for a personalized scenario.

### 6.3 Agent that anticipates user health emergencies

Today's e-health solutions provide important contributions to the health management of the elderly and chronically ill within their own homes. E-health solutions provide for constant monitoring of many vital parameters via portable sensors (inserted into shirts, bracelets, watches, etc.), in order to be able to identify and opportunely signal any hazardous situation requiring intervention. In most cases, however, by the time the call for help is issued, the emergency is already in progress.

One open issue regarding *AmI* is related to recognizing unusual or dangerous situations in order to anticipate health emergencies by monitoring users' habitual activities and capturing their normal behaviour. Such functionalities can be implemented using systems based on machine learning techniques, which exploit artificial intelligence algorithms to learn users' habits by accumulating "experience" on their normal day-to-day activities in order to be able to recognize "abnormal" situations. A system able to anticipate danger before life-threatening situations arise would certainly lead to faster and more effective interventions when used to predict health problems in time and it can thus often save lives.

This agent uses substantially the same algorithms defined in 6.2.1 and 6.2.2 subsections but with some changes. In particular, the procedures that select, learn and apply rules are based taking into account medical recommendations.

Recommendations/Actions	Climb stairs	Rest	Coughing	...	...
Recomm 1	+ 5 %	+ 20 %			
Recomm 2			+ 50 %		
Recomm 3	+ 5 %		+ 20 %		
...					

**Figure 6.6.** Example of medical recommendations

Medical recommendations are digitally represented using a table-like data structure (Figure 6.6). Every row of the table is composed by cells. Every cell contains an empty value or a percentage. The percentage represents the allowed average deviation between the usual measure of the corresponding action and those of a prefixed period. A measure can be a duration of the execution of an action (i.e. a rest or climbing the stairs) or the number of times that an action is performed during a prefixed time interval (i.e. coughing). If all the average deviations in a row are exceeded in the prefixed period, it is recognized as a possible health risk.

When a frequent item reaches the conditions to become a scenario, a rule is created represented via the form  $X \Rightarrow Y$ .  $X$  and  $Y$  are two sets composed both by one or more actions which union represents the corresponding scenario. It is worthwhile underlining that the functioning of the method does not depend on the exact order of execution of the actions that are in the scenario, but it is only considered the temporal correlations existing between the actions performed, since the actions can not always be performed

by the same user in the same order. For the purpose, in fact, it is important how the actions are correlated independently of the order. The only exception is that the actions in X that must be executed by the user before the actions in Y. For example, if there is a scenario where there is the interest to measure the time that the user spends to rest after an effort like climbing the stairs, the rule will be `climb the stairs => sit`. The `climb the stairs` action is important to give a sense to the `sit` action. On the contrary, a rule `rest => climb the stairs` is not significant because the `rest` action is not related to any effort that is previously done. A rule `switch on TV => sit` is also not interesting because the `sit` action can be related to watch the TV and not to a rest.

The decision to learn or discard a scenario and how to partition it in the X and Y sets, is delegated to an ad-hoc algorithm that follows medical recommendations. As example, a reasonable choice is to assign all the actions that require a significant effort, in the X groups.

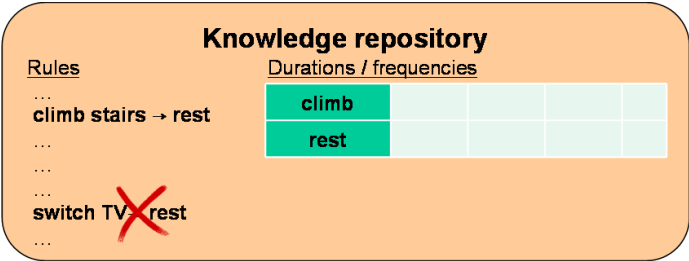


Figure 6.7. Creation of structures in Knowledge repository

At the creation of the scenario, a specific data structure is created (Figure 6.7) in the *Knowledge Repository* module. Every time the user executes the scenario, the structure is filled with durations or frequencies (Figure 6.8).

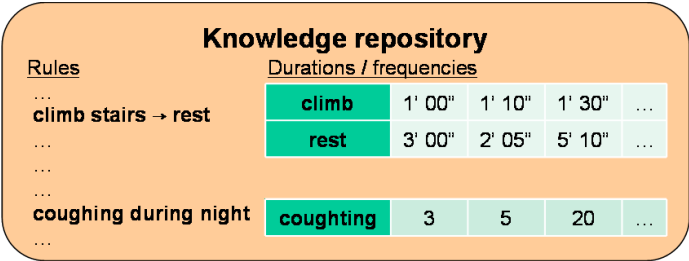


Figure 6.8. Filled structures in Knowledge repository

For the application of the learned scenario, the user has to execute the action belonging to it. The measure of all user's actions is needed because it is not possible to know in advance if the user is actuating a scenario. When we are able to recognize it, part of

the actions that compose that scenario are already performed and it is not possible to get that measures anymore. However, only the measures that compose scenarios are considered and evaluated.

The removal of a scenario comes about when the confidence level does not reach, within a certain period, the pre-set threshold value for triggering the scenario.

Periodically, the *Evaluation Algorithm* verifies if there are the conditions to find in advance signals of possible health problems. If the algorithm verifies the conditions comparing measured values with the medical recommendations, it throws the alarms informing automatically about the possibility that there could be a worsening of the disease and that a medical check could be useful.



## Test and verification

The prototype's functionality has been verified and validated in a use case study aimed at checking the system's ability to learn user's habits, anticipate them and recognize potential hazards to the users' health. The software tests were performed at the *ISTI-CNR* laboratory, where a domotic environment simulating a real residence has been set up (Figure 7.1).



**Figure 7.1.** A view of the ISTI-CNR laboratory

A volunteer member of the research team was assigned the task of interacting with the system during the course of his normal work in the laboratory. During the two-weeks test period the volunteer carried out customary daily habits under usual circumstances, that is, by simply performing a series of repetitive activities within the test setting.

Moreover, a number of actions, suggested by a cardiologist as typical of a potential worsening of a heart patient's state of health, were also simulated. To this end, four pa-

parameters representative of cardiac risk factors were inserted into the algorithm: coughing, using the toilet, rest hours and body weight. The test environment was equipped with the following domotic devices:

- *presence sensors*: to detect when someone enters or leaves a room;
- *dimmer light*: another way to detect the presence of someone in the room;
- *thermostat*: to measure the current ambient temperature;
- *pressure sensor under the sofa and bed*: to detect when someone is sitting or lying down;
- *panic button plus microphone*: to send an alarm and communicate from any room in the house;
- *a fall detection system*: to detect when a user falls;
- *flood, fire and gas sensors*: detecting anomalies in the environment;
- *motion sensors*: to verify if particular rooms in the house are not being used. When rooms such as the bathroom and kitchen remain unused, it may signal the user's inability to fulfill the most basic needs;
- *door opening sensors*: to detect that a person is leaving;
- *a pocket accelerometer*: so that the user's movements throughout the entire day can be followed (time to climb the stairs, use of an exercycle, number and duration of movements around the house), as well as the duration of subsequent rest periods;
- *a microphone was installed in the user's bedroom*: to measure the frequency of coughing, and its recordings were processed with an expressly developed *DSP (Digital Signal Processing)* software;
- *night-time bathroom use*: was checked by simply counting the number of times the bathroom light was turned on;
- *a body scale* to check changes in body weight transmits data to the data control system in real time.

The habitual activities had to be performed in relation to precise circumstances, which were: "waking up in the morning and having breakfast", "going out", "returning home from work in the evening and having supper", and "after supper until going to bed". In order to reduce system training time and the subjects' length of stay in the laboratory, data were collected for a further four weeks using realistic random data with information replication in order to have data available on eight weeks of system use.

At the end of the test, the system validation was carried out according to the *K-Fold Cross Validation* method [76]. The rules produced during the execution of the test are:

- at 7:30 am each day => set thermostat to 21 ° C, switch on bedside lamp, open bedroom blinds, switch on bathroom light;
- successful activation of fingerprint reader => switch on living room light, switch on dining room light, switch on kitchen light, switch on kitchen TV, deactivate intruder alarm, set thermostat to 21 ° C;
- detection of occupant in living room => switch on TV;



- 
- at 11:00 pm each day => close all blinds and switch off all lights, close water and gas electromagnetic valves, activate intruder alarm, set thermostat to 18° C;
  - preferred temperatures 11:00 pm–07:30 am => 18°;
  - preferred temperatures 07:30 am–11:00 pm => 21°;
  - preferred music upon reentry => Jazz;
  - preferred music on awakening => Classical;
  - coughing + 30%, toilet use +20%, rest hours +34% => Alarm;
  - using the toilet +30%, body weight +0.5% => Alarm;
  - rest hours +21%, toilet use +85%; body weight +0.3% => Alarm;

After the test period, the software was able to learn sufficient user baseline routines. Regarding the ability to anticipate user's actions, the best results were obtained by setting the correlation window size to fifteen minutes, while the best setting for successful prediction of a possible health hazard was seven days. With such a value, the validation performed using the available dataset yielded a specificity value of 89%, with 88% sensitivity and 89% accuracy.



## Conclusions and future works

Apart from making life more comfortable for users without particular needs, the *software system* can offer significant advantages to the elderly and/or disabled people as well. For these segments of the population, even the simplest of everyday's actions may represent an insurmountable obstacle, hence a system that learns their habits and performs actions in their stead can offer much needed support and safeguards. It can moreover contribute to reducing the currently acute problem of the digital divide.

This system can also provide advantages in terms of energy savings. Once the system has acquired and learned good practices such as switching off lights and closing shutters depending on the time of day or in relation to a change in season, this information will never be lost and subsequent automatic system actions will be able to contribute significantly to saving energy.

The results obtained serve to illustrate the effectiveness of the approach adopted and its potential for use in cooperation and interoperability with all existing domotic systems without the use of a specialized software for each type of device.

The system developed fits well within the perspective and goals of current *Ambient Intelligence* research. The implementation of the automatic learning system is able to learn types of habits, which enable covering a large part of the common behaviors of *Ambient Intelligence* system users, and can anticipate such users' behaviors and recognize critical health situations in advance quite reliably. In fact, software validation test performed has also demonstrated that this tool can be an important aid for preventing any possible deterioration of the user's state of health, as it already supports the application of suitable rules and thresholds for recognizing any changes in some user's behaviors or sensor parameters that would signal a potential rise in any of the risk factors. In any event, the need for the help and cooperation of medical specialists will undoubtedly be essential for enabling the system to actually be used as a medical prevention tool.

With regard to related work following similar approaches, none seems able to cover all the aspects considered by the developed *Intelligent Agents*. The choice of following a hybrid approach - applying both the *Data Mining* technique of *associative rule* learning and *statistical learning* methods - has rendered the system more versatile and reliable. In

fact, combining the two forms of learning has led to a summing of the strong points of the two different learning methods, while limiting their respective shortcomings. Moreover, system's performance improves over time, as new experience was accumulated. The number of errors committed by the system was relatively low right from the start and then fell further as the system acquired ulterior data.

However, as it can be expected, in order to achieve more realistic results the prototype clearly requires more thorough, longer-duration testing in order to improve learning and enable more careful evaluation of the system's parameters. A more extensive dataset will surely enable more accurate validation and evaluation of the system's capabilities. The system thus represents a good point of departure for future development, with the main goal of improving the learning capacity achieved so far.

Summarising, the main original results obtained in this work are:

- *the development of a platform for interoperability between heterogeneous automation systems based on a distributed architecture type Digital Ecosystem*: the study [73] [60] has led to the definition of an innovative model of representation of reality housing conforms to the vision of *Ambient Intelligence*. The *Digital Ecosystem* approach was the most suitable to the circumstances of design as it has enabled the implementation of a collaborative system between the entities involved;
- *the development of a scalable, oriented to the Internet of Things*: The study in the *Internet of Things* field has helped to provide answers to some of the open questions in the field of research in the area, including the need to find solutions that allow direct interfacing of home automation devices, with the *IoT* world. The proposed solution [57] is scalable, universal and based on established standards. It is lowered into reality; takes account of the technologies and home automation systems on the market today and the limits of the possible hardware and software devices to be interfaced;
- *the design and implementation of a comprehensive ontological model able to describe the environment, the user and the device characteristics and their interaction model in order to implement interoperability*: the study has allowed to obtain a solution for interoperability [59] between the domotic systems implemented at the semantic level, by exploiting the ability of reasoning typical of an ontological approach [56];
- *the definition of a new methodology of Artificial Intelligence hybrid applied in the fields of environmental comfort and the Ambient Assisted Living*: the study has allowed the development of an instrument of Machine Learning able to learn the habits of the user. The instrument is able to learn both usual actions, as the usual habitat preferences [56] of the inhabitants. In addition, using the same techniques, a new algorithm designed to anticipate possible deterioration of health conditions was developed[58].

## 8.1 Future works

In order to achieve sufficiently reliable deductions, the data acquired through the environmental sensors are alone not sufficient. Such data must be integrated with the information that can be captured via wearable devices able to provide data about:

- body temperature;
- blood pressure;
- pulse;
- blood oxygen saturation;
- body weight;
- percentage of body fat;
- percentage of muscle;
- percentage of water;
- blood glucose;
- EKG;
- peak of expiratory flow;
- coagulation.

Considering the wearable sensors that may be added, possible candidates for testing will be patients that meet the following criteria:

- *Chronic Disease History*: primary diagnosis of *Chronic Obstructive Pulmonary Disease (COPD)* and / or Chronic Heart trouble; one or more hospital emergency admission in the last year due to exacerbation of the chronic disease and unstable conditions deemed to be due to anxiety about their condition;
- *Caretaker functioning/ability*: reasonable cognitive ability to report observations and reasonable skills in using the home appliances and peripherals for vital signal measurements (e.g. blood pressure, pulse, glucose, oxygen saturation, and body weight and temperature).

The goal of testing as applied to both actual scenarios and patients is to verify:

- whether the system can be considered a useful, convenient tool for health care delivery;
- whether it will be able to save time and money by reducing hospital admissions, emergency room and medical practitioner visits and associated travel;
- that users feel they are better informed about their health conditions, thus promoting active participation in their health management and empowering them to perform better self-care;
- that the system can improve health management by providing physicians with more accurate, up-to-date information to help them make better decisions.

The awareness of the elderly people and their reactions to being continually monitored and supervised may present obstacles. They will likely feel controlled and managed by something that they do not fully understand and must thus be given a sense of security and protection, without inducing anxiety [62].

Exploiting the use of the developed ontologies, the *software platform* furnishes the basis for the creation of a friendly human-machine interface (HMI) based on the use of natural language, to:

- make the system autonomously able to know the actions to perform knowing the goal to reach (e.g. “have more light in living room” the system can turn on a lamp or open the blinds);
- pinpoint domotic devices using relative positioning (e.g. lamp above the table);
- refer to home environments using their characterizations (e.g. to refer to the kitchen, the user can say: “the room where I cook”);

Finally, further important future work will be dedicated to enabling the system to identify specific users and their locations, for instance when they enter or leave a room. To improve such identification, an RFID-based strategy [16] [20] can be employed to enable the system to recognize who performs an action, when and where.

---

## References

1. Emile Aarts and Boris de Ruyter. New research perspectives on ambient intelligence. *Journal of Ambient Intelligence and Smart Environments*, 1(1):5–14, 2009.
2. R. Aggarwal, K. Verma, J. Miller, and W. Milnor. Constraint driven web service composition in meteor-s. In *Services Computing, 2004. (SCC 2004). Proceedings. 2004 IEEE International Conference on*, pages 23–30, 2004.
3. Rama Akkiraju, Joel Farrell, John Miller, Meenakshi Nagarajan, Marc-Thomas Schmidt, Amit Sheth, and Kunal Verma. Web service semantics-wsdl-s. *W3C member submission*, 7, 2005.
4. Jan Alexandersson. i2home—towards a universal home environment for the elderly and disabled. *Künstliche Intelligenz*, 8(3):66–68, 2008.
5. Jesús M. Almendros-Jiménez. A prolog library for owl rl. In *Proceedings of the 4th International Workshop on Logic in Databases, LID '11*, pages 49–56, New York, NY, USA, 2011. ACM.
6. Deepak Alur, Dan Malks, John Crupi, Grady Booch, and Martin Fowler. *Core J2EE Patterns (Core Design Series): Best Practices and Design Strategies*. Sun Microsystems, Inc., 2003.
7. Grigoris Antoniou and Frank Van Harmelen. Web ontology language: Owl. In *Handbook on ontologies*, pages 67–92. Springer, 2004.
8. Ken Arnold, Robert Scheifler, Jim Waldo, Bryan O'Sullivan, and Ann Wollrath. *Jini Specification*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1999.
9. Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.
10. Fabio Bellifimone, Giovanni Caire, Agostino Poggi, and Giovanni Rimassa. Jade—a white paper. *EXP in search of innovation*, 3(3):6–19, 2003.
11. Tim Berners-Lee, James Hendler, Ora Lassila, et al. The semantic web. *Scientific american*, 284(5):28–37, 2001.
12. H. Boley and E. Chang. Digital ecosystems: Principles and semantics. In *Digital EcoSystems and Technologies Conference, 2007. DEST '07. Inaugural IEEE-IES*, pages 398–403, 2007.
13. H. Boudali, P. Crouzen, B.R. Haverkort, M. Kuntz, and M. I A Stoelinga. Architectural dependability evaluation with arcade. In *Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on*, pages 512–521, 2008.
14. G. Briscoe and A. Marinos. Digital ecosystems in the clouds: Towards community cloud computing. In *Digital Ecosystems and Technologies, 2009. DEST '09. 3rd IEEE International Conference on*, pages 103–108, 2009.
15. Alan W. Brown. Model driven architecture: Principles and practice. *Software and Systems Modeling*, 3(4):314–327, 2004.
16. D. Bruckner, C. Picus, R. Velik, W. Herzner, and G. Zucker. Hierarchical semantic processing architecture for smart sensors in surveillance networks. *Industrial Informatics, IEEE Transactions on*, 8(2):291–301, May 2012.

17. Silvana Castano, Alfio Ferrara, and Stefano Montanelli. Ontology-based interoperability services for semantic collaboration in open networked systems. In Dimitri Konstantas, Jean-Paul Bourrières, Michel Léonard, and Nacer Boudjlida, editors, *Interoperability of Enterprise Software and Applications*, pages 135–146. Springer London, 2006.
18. PG Censoni, Piero De Sabbata, Guido Cucchiara, Fabio Vitali, Luca Mainetti, and Thomas Imolesi. Moda-ml, a vertical framework for the textile-clothing sector based on xml and soap. *Challenges and achievements in e e-business and e-work*, 2002.
19. B. Chandrasekaran, John R. Josephson, and V. Richard Benjamins. What are ontologies, and why do we need them? *IEEE Intelligent Systems*, 14(1):20–26, January 1999.
20. Yuan-Hsin Chen, Shi-Jinn Horng, Ray-Shine Run, Jui-Lin Lai, Rong-Jian Chen, Wei-Chih Chen, Yi Pan, and T. Takao. A novel anti-collision algorithm in rfid systems for identifying passive tags. *Industrial Informatics, IEEE Transactions on*, 6(1):105–121, Feb 2010.
21. DBE Consortium. Dbe studio.
22. DBE Consortium. Evolutionary environment habitat network.
23. DBE Consortium. Sbeaver - business modeller editor.
24. DBE Consortium. Dbe servent, June 2007.
25. Oasis Consortium. ebxml - enabling a global electronic market.
26. OASIS Consortium. Oasis advancing open standards for the information society.
27. OASIS Consortium. Oasis ebxml collaboration protocol profile and agreement (cppa) tc.
28. OASIS Consortium. Oasis ebxml messaging services tc.
29. Shoenet Consortium. Shoenet.
30. Alan Cooper and Robert Reimann. *About Face 2.0: The Essentials of Interaction Design*. Books24x7. com, 2005.
31. EDI. Edi - electronic data interchange.
32. Edmonds. D26.7: Dbe portal specification. Technical report, Intel, 2005.
33. Bob Emmerson. M2m: the internet of 50 billion devices. *WinWin Magazine*, pages 19–22, 2010.
34. P Ferronato. D21. 2 architecture scope document. internal, 2005.
35. P. Ferronato. Architecture for digital ecosystems, beyond service oriented architecture (ieee-dest 2007). In *Digital EcoSystems and Technologies Conference, 2007. DEST '07. Inaugural IEEE-IES*, pages 660–665, 2007.
36. Christian Fuchs. *Internet and society: Social theory in the information age*. Routledge, 2007.
37. Sarah Gillinson, Hannah Green, and Paul Miller. *Independent Living: The right to be equal citizens*. Demos, 2005.
38. Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2):199–220, June 1993.
39. gtin. Gtin info.
40. Nicola Guarino. Formal ontology and information systems. In *Proceedings of the 1st International Conference June 6-8, 1998, Trento, Italy*, pages 3–15. IOS Press, 1998.
41. Sten Hanke, Christopher Mayer, Oliver Hoeffberger, Henriette Boos, Reiner Wichert, Mohammed-R. Tazari, Peter Wolf, and Francesco Furfari. universaal – an open and consolidated aal platform. In Reiner Wichert and Birgid Eberhardt, editors, *Ambient Assisted Living*, pages 127–140. Springer Berlin Heidelberg, 2011.
42. Hans-Jörg Happel and Stefan Seedorf. Applications of ontologies in software engineering. In *International Workshop on Semantic Web Enabled Software Engineering (SWESE'06)*, November 2006.
43. Russell Hardin. The free rider problem (stanford encyclopedia of philosophy). <http://plato.stanford.edu/entries/free-rider/>, 2003. [Online; accessed 17-May-2013].
44. Jim Hendler. Web 3.0: Chicken farms on the semantic web. *Computer*, 41(1):106–108, January 2008.
45. Matthew Horridge and Sean Bechhofer. The owl api: A java api for owl ontologies. *Semantic Web*, 2(1):11–21, 2011.



46. UBL Italia. Ubl-italia.
47. Antonio J Jara, Pedro Moreno-Sanchez, Antonio F Skarmeta, Socrates Varakliotis, and Peter Kirstein. Ipv6 addressing proxy: Mapping native addressing from legacy technologies and devices to the internet of things (ipv6). *Sensors*, 13(5):6687–6712, 2013.
48. Suho Jeong, Seong Hoon Kim, Minkeun Ha, Taehong Kim, Jinyoung Yang, Nam Giang, and Daeyoung Kim. Enabling transparent communication with global id for the internet of things. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*, pages 695–701. IEEE, 2012.
49. Markus Jung, Christian Reinisch, and Wolfgang Kastner. Integrating building automation systems and ipv6 in the internet of things. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*, pages 683–688. IEEE, 2012.
50. Markus Jung, Jürgen Weidinger, Wolfgang Kastner, and Alex Olivieri. Heterogeneous device interaction using an ipv6 enabled service-oriented architecture for building automation systems. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, pages 1939–1941. ACM, 2013.
51. Lee Kennedy. D24.7: Distributed storage system. Technical report, Intel, 2005.
52. S. Lorente. Key issues regarding domotic applications. In *Information and Communication Technologies: From Theory to Applications, 2004. Proceedings. 2004 International Conference on*, pages 121–122, 2004.
53. Jeff McAffer, Jean-Michel Lemieux, and Chris Aniszczczyk. *Eclipse rich client platform*. Addison-Wesley Professional, 2010.
54. GS1 Members. Global location number.
55. GS1 Members. Gs1 - the global language of business.
56. Vittorio Miori and Dario Russo. Anticipating health hazards through an ontology-based, iot domotic environment. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*, pages 745–750. IEEE, 2012.
57. Vittorio Miori and Dario Russo. Domotic evolution towards the iot. In *Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference on*, pages 809–814. IEEE, 2014.
58. Vittorio Miori and Dario Russo. Preventing health emergencies in an unobtrusive way. In *International Conference on IoT Technologies for HealthCare (HealthyIoT), 2014 First International Conference on*. IEEE, 2014.
59. Vittorio Miori, Dario Russo, and Cesare Concordia. Meeting people's needs in a fully interoperable domotic environment. *Sensors*, 12(6):6802–6824, 2012.
60. Pillitteri L. Russo D. Miori V., Bianchi Bandinelli R. Research ambient assisted living solutions in the european context. Technical report, Institute of Science and Technologies of Information (ISTI) - National Research Council of Italy (CNR), 2014.
61. moda ml. moda-ml.
62. Stuart Moran and Keiichi Nakata. Ubiquitous monitoring and user behaviour: A preliminary model. *J. Ambient Intell. Smart Environ.*, 2(1):67–80, January 2010.
63. Sudhakiran V Mudiam, Gerald C Gannod, and Timothy E Lindquist. Synthesizing and integrating legacy components as services using adapters. *Science of Computer Programming*, 60(2):134–148, 2006.
64. Shelena Soosay Nathan, Sanjaav Selan Mohan, Adelin Rose Harudas, and Kashif Nisar. Berkeley internet name domain (bind).
65. OMG. Semantics of business vocabulary and business rules specification, second interim specification, 2006.
66. Kotopoulos Pappas, Kotopoulos. D14.5: Final p2p implementation of the dbc knowledge base and sr. Technical report, Technical University of Crete, 2005.
67. Abhijit A. Patil, Swapna A. Oundhakar, Amit P. Sheth, and Kunal Verma. Meteor-s web service annotation framework. In *Proceedings of the 13th international conference on World Wide Web, WWW '04*, pages 553–562. ACM, 2004.

68. Konstantinos Perakis, Gianna Tsakou, Christoforos Kavvadias, and Alkis Giannakoulis. Homedotold, home services advancing the social interaction of elderly people. In José Bravo, Ramón Hervás, and Vladimir Villarreal, editors, *Ambient Assisted Living*, volume 6693 of *Lecture Notes in Computer Science*, pages 180–186. Springer Berlin Heidelberg, 2011.
69. Joanne H Pratt. *E-Biz. com: Strategies for small business success*. SBA Office of Advocacy, 2002.
70. Preeda Rajasekaran, John Miller, Kunal Verma, and Amit Sheth. Enhancing web services description and discovery to facilitate composition. In *Semantic Web Services and Web Process Composition*, pages 55–68. Springer, 2005.
71. Trygve Mikjel H Reenskaug. The original mvc reports. 1979.
72. James Robertson and Suzanne Robertson. Volere: Requirements specification template. Technical report, Technical Report Edition 6.1, Atlantic Systems Guild, 2000.
73. Dario Russo. Domotics and robotics. In *The Church at the Service of Sick Elderly People: Care for People with Neurodegenerative Pathologies*, 2013.
74. Michele Ruta, Floriano Scioscia, Giuseppe Loseto, and Eugenio Di Sciascio. An agent framework for knowledge-based homes. In *3rd International Workshop on Agent Technologies for Energy Systems (ATES 2012)(2012, to appear)*, 2012.
75. Amit Sheth. Semantic web process lifecycle: role of semantics in annotation, discovery, composition and orchestration, 2003.
76. Takahiro Shinozaki and Mari Ostendorf. Cross-validation and aggregated em training for robust parameter estimation. *Comput. Speech Lang.*, 22(2):185–195, April 2008.
77. Andrew Sixsmith, Sonja Meuller, Felicitas Lull, Michael Klein, Ilse Bierhoff, Sarah Delaney, and Robert Savage. Soprano – an ambient assisted living system for supporting older people at home. In Mounir Mokhtari, Ismail Khalil, Jérémy Bauchet, Daqing Zhang, and Chris Nugent, editors, *Ambient Assistive Health and Wellness Management in the Heart of the City*, volume 5597 of *Lecture Notes in Computer Science*, pages 233–236. Springer Berlin Heidelberg, 2009.
78. Nikolaos Spanoudakis, Pavlos Moraitis, and Yannis Dimopoulos. Engineering an agent-based approach to ambient assisted living. In *Adjunct Proceedings of the 3rd European Conference on Ambient Intelligence (Aml09), Workshop on Interactions Techniques and Metaphors in Assistive Smart Environments (IntTech'09), Salzburg, Austria*, pages 268–271. Citeseer, 2009.
79. Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.
80. Mohammad-Reza Tazari, Francesco Furfari, Juan-PabloLázaro Ramos, and Erina Ferro. The persona service platform for aal spaces. In Hideyuki Nakashima, Hamid Aghajan, and Juan-Carlos Augusto, editors, *Handbook of Ambient Intelligence and Smart Environments*, pages 1171–1199. Springer US, 2010.
81. TechIDEAS Asesores Tecnológicos. fada - federated advanced directory architecture, 2007.
82. Texweave. Texweave - standardization and interoperability in the textile supply chain integrated.
83. Tania Tudorache, Csongor Nyulas, Natalya Fridman Noy, and Mark A. Musen. Webprotégé: A collaborative ontology editor and knowledge acquisition tool for the web. *Semantic Web*, 4(1):89–99, 2013.
84. L. van Moergestel, W. Langerak, and J.-J. Meyer. Agents in domestic environments. In *Control Systems and Computer Science (CSCS), 2013 19th International Conference on*, pages 487–494, 2013.
85. Jorge E. López De Vergara, Víctor A. Villagrà, and Julio Berrocal. Semantic management: advantages of using an ontology-based management information meta-model. In *Proceedings of the HP Openview University Association Ninth Plenary Workshop (HP-OVUA'2002), distributed videoconference*, pages 11–13, 2002.
86. Jean-Baptiste Waldner and Jean Baptiste Waldner. *Nano-informatique et intelligence ambiante: inventer l'ordinateur du XXle siècle*. Hermès science publications, 2007.

87. Zhiqiang Wei, Jing Li, Yongquan Yang, and Dongning Jia. A residential gateway architecture based on cloud computing. In *Software Engineering and Service Sciences (ICSESS), 2010 IEEE International Conference on*, pages 245–248. IEEE, 2010.
88. Mark Weiser. Human-computer interaction. chapter The Computer for the 21st Century, pages 933–940. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995.
89. Tin-Yu Wu, Chia-Chang Hsu, and Han-Chieh Chao. Ipv6 home network domain name auto-configuration for intelligent appliances. *Consumer Electronics, IEEE Transactions on*, 50(2):491–497, 2004.
90. Tin-Yu Wu, Chia-Chang Hsu, and Han-Chieh Chao. Ipv6 home network domain name auto-configuration for intelligent appliances. *Consumer Electronics, IEEE Transactions on*, 50(2):491–497, 2004.



---

## Index

- Address Resolution Protocol, 73
- Ambient Assisted Living, 4
- Ambient Intelligence, 1–3, 6, 91, 115, 116
- Artificial Intelligence, 13, 19
- Assistive Technologies, 5
- Auto IP, 72
- Automatic Private IP Addressing, 72
- Automatic Speech Recognition, 93
- Biological Ecosystem, 9
- bit, 75
- Building Automation, 11
- Class, 17
- Common Object Request Broker
  - Architecture, 74
- Concept, 18
- Conceptualisation, 13
- DAML+OIL, 16
- Data Mining, 7–9, 19, 20
- DBE project, 38
- Declarative Knowledge, 13
- Description Logic, 17
- DHCP Server, 72
- Digital Ecosystem, 4, 9, 10, 38, 61, 116
- DogOnt, 50
- Domain of Interest, 13
- Domotic Agent, 71, 77–80, 85, 86, 92
- domotic bus, 77
- Domotics, 11
- Domotique, 11
- Dynamic Host Configuration Protocol, 72
- eBiz-TCF, 44
- Evaluation Algorithm, 109
- Free Riding, 10
- Home Automation, 2, 11
- Human Machine Interface, 92
- HyperText Transfer Protocol, 74
- Independent Living, 6
- Individual, 9
- Instance, 17
- Internet Of Objects, 6
- Internet Of Things, 3, 6, 12, 13
- Internet Printing, 73
- Internet Printing Protocol, 73
- IP Address, 72, 73
- IP Multicasting, 73
- IPP, 73
- IPv4, 12
- IPv6, 12
- Java RMI, 74
- KNX, 71
- Loom, 16

Machine Learning, 19, 20  
 Metadata, 16, 17  
 Multicast Address, 73  
  
 Name, 17, 18  
  
 OIL, 17  
 OKBC, 18  
 Ontologies, 13, 14, 16  
 Ontology, 13, 14, 16  
 Ontology Inference Layer, 17  
 Ontology Web Language, 16, 17  
 Ontosphere3d, 19  
 OntoViz, 19  
 Open Knowledge Base Connectivity, 18  
 Open Source, 18  
 OWL, 16–19  
 OWL-DL, 18  
 OWL-Full, 18  
 OWL-Lite, 18  
  
 Plug and Play, 71  
 Port Number, 73  
 Predicate, 17  
 Prolog, 16  
 Property, 17  
 Protégé, 18, 19, 68, 69  
 Protégé Plugin Library, 19  
  
 RDF, 16, 17  
 RDF Data Model, 16, 17  
 RDF Document, 17  
 RDF Expression, 16  
 RDF Syntax, 17  
 RDFS, 17  
 Real Time Streaming Protocol, 74  
 Relation, 18  
 Remote Method Invocation, 74  
 Resource, 16, 17  
 Resource Description Framework, 16  
 Role, 18  
  
 RTSP, 74  
  
 Semantic Intelligence, 3  
 Semantic Reasoner, 13  
 Semantic Web, 13, 19  
 Sensor Model Language, 51  
 Service Oriented Architecture, 10  
 Simple Service Discovery Protocol, 73  
 Slot, 18  
 Smart Home, 11  
 Software Platform, 89, 93  
 Statement, 17  
 Subject, 17  
  
 TCP-IP Stack, 72  
 Text To Speech, 92  
  
 Ubiquitous Computing, 1  
 UDP Packet, 73  
 Uniform Resource Locator, 16  
 Universal Description Discovery and  
     Integration, 10  
 Universal Plug and Play, 71  
 Universal Resource Identifier, 16  
 UPnP Architecture, 72  
 UPnP Device, 71  
 UPnP Device Architecture, 72  
 UPnP Forum, 72  
 UPnP Network, 71  
 URI, 16, 73  
 URL, 16  
 User Agent, 89  
  
 Value, 17  
  
 Web 3.0, 3  
 Web Services Description Language,  
     10  
  
 XML, 16, 17, 72, 74  
 XML Encoding, 17  
 XML Tag, 17

---

## Acknowledgments

Foremost, I would like to thank my advisors Dr Vittorio Miori and Prof. Stefano Giordano for their advices and their guidance during these years and for giving me the opportunity to work in this beautiful research area.

Besides my advisors, I would also like to express my most sincere gratitude to my parents, to my grandparents and to my wife Ada, who have always trusted and supported me during my studies, my work and my life.

Last but not least, I would like to thank my closest friends. In particular I would like to thank Marco Righi for being my dear friend as well as good colleague at work and companion during these studies, and Luca Saiu for teching me a lot of technical notions and, expecially, the true essence of being a computer scientist.